НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислюваної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК _____

«До захисту допущено»
В.о. завідувача кафедри
_____ О.І. Ролік
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Сервіс тематичного аналізу інформації для платформи Телеграм»

Виконав:
студент ІІ курсу, групи ІА-61м
Смолинець Остап Тарасович _____

Керівник:
Доцент, к.т.н., доцент,
Катін, П.Ю. _____

Рецензент:
Доцент, к.т.н., доцент,
Радько, І.П. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислюваної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою
Спеціальність – 151 «Автоматизація та комп'ютерно інтегровані технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С.Ф., Теленик
«___»_____20_ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Смолинцю Остапу Тарасовичу

1. Тема дисертації «Сервіс тематичного аналізу інформації для платформи Телеграм», науковий керівник дисертації Катін Павло Юрійович, к. т. н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – засоби миттєвого обміну повідомленнями.

4. Предмет дослідження – можливості використання нейронних мереж у ботах для месенджерів.

5. Перелік завдань, які потрібно розробити: описання загальних принципів створення сервісів для сучасних месенджерів; описання можливості використання нейронних мереж для покращення досвіду користувача, при взаємодії з ботом; розробка сервісу для платформи Телеграм; написання пояснювальної записки.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: графік кількості активних користувачів у найпопулярніших месенджерах, графік росту кількості активних користувачів за місяць у месенджері Телеграм, алгоритм вибору засобів розробки бота.

7. Перелік публікацій: 1) «Можливість використання нейронних мереж у системах обміну миттєвими повідомленнями» – Науковий журнал «Наука Онлайн» – 2018 – №5(5) – С.27 – 32; 2) «Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data» – VI Міжнародна науково-практична конференція «Winter Infocom Advanced Solutions 2017»

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд сучасних месенджерів	15.03.18 – 29.03.18	
2	Аналіз лідерів у сфері миттєвого обміну повідомленнями	30.03.18 – 07.04.18	
3	Аналіз можливостей використання ботів	08.04.18 – 13.04.18	
4	Порівняння архітектури ботів із відкритим кодом	14.04.18 – 17.04.18	
5	Дослідження можливості застосування засобів нейронних мереж	18.04.18 – 22.04.18	
6	Розробка чат бота	23.04.18 – 02.05.18	
7	Оформлення пояснювальної записки	03.05.18 – 14.05.18	

Студент О.Т., Смолинець

Науковий керівник дисертації П.Ю., Катін

РЕФЕРАТ

Магістерська дисертація освітньо-кваліфікаційного рівня “магістр” на тему “Сервіс тематичного аналізу інформації для платформи Телеграм”: 113с., 31 рис., 24 табл., 3 додатки, 38 джерела.

Об'єкт дослідження – засоби миттєвого обміну повідомленнями.

Мета роботи – розробити сервіс для платформи Телеграм, який використовуватиме описані у роботі ключові принципи створення ботів для месенджерів. Застосувати засоби нейронних мереж для покращення досвіду користувача при використанні бота.

Системи миттєвого обміну повідомленнями використовуються сотнями міліонів людей по всьому світу. Вони перетворилися із засобів для спілкування між людьми у засоби для отримання інформації та у неймовірно потужний маркетинговий інструмент. Ключовою частиною, при їх використанні, є можливість спілкуватися із ботами. Тому у цій дисертації розглядаються основні принципи розробки ботів та можливості використання нейронних мереж для максимізації позитивного досвіду користувача. У дисертації наведено алгоритм вибору необхідних інструментів для розробки та приклади їх поєднання.

Для розробки використовувалася мова програмування C# 6 та засоби ASP.NET.

МЕСЕНДЖЕРИ, СИСТЕМИ МИТТЄВОГО ОБМІНУ ПОВІДОМЛЕННЯМИ, ЧАТ БОТ, БОТ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ПРИРОДНЬОЇ МОВИ, ШТУЧНИЙ ІНТЕЛЕКТ.

ABSTRACT

Master thesis "Thematic analysis of information for the Telegram platform": 113s, 31 picture, 24 tables, 3 appendices, 38 sources.

Object of study – means of instant messaging.

The purpose of the work is to develop a service for the Telegram platform, which will use the key principles of messenger bots creation described in the work. Apply neural networks to improve the user experience when using the bot.

Hundreds of millions of people around the world use instant messaging systems. They have become not only the primary means of communication between people but also the means of obtaining information and an incredibly powerful marketing tool. One of the key features of their use is the ability to communicate with bots. Therefore, this thesis examines the basic principles of bots development and the possibility of using neural networks to enhance user experience. The thesis presents an algorithm for selecting the necessary tools for development and some examples of their combination.

Programming language C # 6 and ASP.NET tools were used for development.

MESSENGERS, INSTANT MESSAGING SYSTEMS, CHAT BOT, BOT, NEURAL NETWORKS, NATURAL LANGUAGE PROCESSING, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ОДИНИЦЬ І ТЕРМІНІВ

RSS – Rich Site Summary (Короткий Зміст Сайту)

СМОП – Система Миттєвого Обміну Повідомленнями

ЧБ – Чат Бот

AIM – AOL Instant Messenger (Месенджер Розроблений Компанією AOL)

API – Application Programming Interface

ППІ – Прикладний Програмний Інтерфейс

B2B – Business-To-Business (Бізнес Для Бізнесу)

B2C – Business-To-Consumer (Бізнес Для Користувача)

ШНМ – Штучна Нейронна Мережа

ReLU – Rectified Linear Unit (Тип Функції Активації)

SGD – Stochastic Gradient Descent (Метод Стохастичного Градієнта)

NER – Named-Entity Recognition (Розпізнавання Названого Суб'єкта)

NLP – Natural-Language Processing (Обробка Природних Мов)

CLR – Common Language Runtime (Загальне Середовище Виконання)

IDE – Integrated Development Environment (Інтегроване Середовище Розробки)

CI – Continuous Integration (Неперервна Інтеграція)

CD – Continuous Delivery (Неперервна Доставка)

QE – Quality Engineer (Інженер з Якості)

QA – Quality Assurance (Забезпечення Якості)

DevOps – Development and Operations (Розробка та Операції)

BA – Business Analytic (Бізнес Аналітик)

PO – Product Owner (Власник Продукту)

GPL – General Public License (Загальна Публічна Ліцензія)

ВСТУП

Останні декілька років популярність систем обміну миттєвими повідомленнями (месенджерів) тільки зростає. Вони перетворилися із засобів для спілкування між людьми у засоби для отримання інформації та у неймовірно потужний маркетинговий інструмент. Чималу роль у цьому відіграли боти. Зараз існує безліч їх варіацій – від ботів для отримання RSS розсилок до ботів для замовлення їжі. Звісно таким корисним інструментом цікавляться власники бізнесу. І хоча боти уже декілька років активно використовуються закордоном, в Україні вони не так поширені та тільки надувають популярності, а отже є перспективною нішею для розробників. Слід враховувати, що для потенційних клієнтів важливим аспектом є досвід користувача при роботі із ботом, який включає те як бот сприймає команди, наскільки добре він виділяє суть запиту користувача та наскільки доречні і зрозумілі його відповіді.

У дисертації описуються загальні принципи створення бота для месенджера, його можлива архітектура, а також розглядається можливість введення функціоналу нейронних мереж для покращення досвіду користувача при використанні бота.

Актуальність теми полягає у необхідності описати загальні принципи створення ботів, як сервісів для сучасних месенджерів, оскільки вони широко застосовуються компаніями різного ґатунку для розвитку їхнього бізнесу. Особливо це актуально для ринку України, оскільки компанії тільки починають освоювати дану технологію, а отже попит на спеціалістів з розробки таких сервісів стрімко зростає.

Мета і задачі дослідження: описати ключові принципи створення ботів для месенджерів. Розглянути можливості застосування засобів нейронних мереж для покращення досвіду користувача при використанні бота. На основі описаних принципів розробити сервіс для платформи Телеграм.

Об’єкт дослідження – засоби миттєвого обміну повідомленнями.

Предмет – боти, як засіб взаємодії із користувачем.

У роботі було досліджено та описано загальні принципи створення бота для месенджера, на прикладі декількох ботів, код яких знаходяться у загальному доступі. За результатами дослідження було виділено можливу архітектуру сервісу. Також

було досліджено можливість введення функціоналу нейронних мереж на прикладі застосування Microsoft Cognitive Services для покращення виведення інформації ботом.

Наукова новизна одержаних результатів

Полягає у описанні загальних принципів побудови ботів для сучасних месенджерів.

Практичне значення одержаних результатів

Полягає у розробці сервісу для платформи Телеграм та використанні Microsoft Cognitive Services для покращення виведення інформації ботом. Полягає у можливості застосування мінімально модифікованої розробки для покращення взаємодії різних компаній із клієнтами.

Апробація результатів роботи

Розроблений сервіс тематичного аналізу інформації для платформи Телеграм було протестовано. Основні положення дисертації було оприлюднено на конференції «WINTER INFOCOM ADVANCED SOLUTIONS 2017», де їх було обговорено та отримано загальне схвалення.

Публікації

Основні положення магістерської дисертації опубліковано у статті «Можливість використання нейронних мереж у системах обміну миттєвими повідомленнями» у науковому журналі «Наука Онлайн». Також було опубліковано статтю «Розробка архітектури системи автоматизованого збору, обробки та аналізу даних на основі технології Big Data» на конференції «WINTER INFOCOM ADVANCED SOLUTIONS 2017». Також тезах конференцій опубліковані результати магістерської роботи.

1 МЕСЕНДЖЕРИ ЯК СУЧАСНИЙ ЗАСІБ СПІЛКУВАННЯ

1.1 Історія розвитку засобів обміну миттєвими повідомленнями з використанням інформаційних технологій

Загальнодоступність комп’ютерів та смартфонів стала поштовхом до розвитку нових сервісів зв’язку. На зміну телефонним дзвінкам прийшли онлайн голосові та навіть відео дзвінки. Люди почали використовувати комп’ютери та інтернет для спілкування. Першим інструментом для цього стала електронна пошта, яка ідейно наслідувала традиційні листи, проте була значно ефективнішою. Далі популярності набули чати. Вони використовувалися для групового спілкування між двома та більше користувачами. Першим багатокористувацьким онлайн чатом був Talkomatic. Він був створений Дугом Брауном і Девідом Р. Уоллі в 1973 році в системі Programmed Logic for Automated Teaching Operations (PLATO, укр. Програмний алгоритм для автоматизованих операцій викладання – перша система для електронного навчання) в Університеті штату Іллінойс [1]. Він запропонував шість каналів (аналог «кімнати»), кожен з яких міг вмістити до п'яти учасників. Звісно, з того часу чати еволюціонували і тепер практично у кожного сервісу, який надає послуги онлайн чату, не існує обмежень на кількість користувачів та на кількість каналів у яких бере участь користувач. Також сучасні чати усе частіше використовують, раніше вузько направлені технологій, такі як нейронні мережі, штучний інтелект та інші. Усі ці нововведення, так чи інакше, направлені і на покращення взаємодії із користувачем. Паралельно з чатами користувачі починали використовувати месенджери для спілкування між собою. Примітивні комп’ютерні системи для передачі даних були розроблені ще у 1970-х роках.

Системи які базувалися на використанні графічного інтерфейсу користувача для відображення отриманого повідомлення набували популярності у 1990-х роках. До перших таких месенджерів належали PowWow, ICQ (рисунок 1) та AOL Instant Messenger (рисунок 2). У той же час інші компанії розробили власне програмне забезпечення для обміну миттєвими повідомленнями (Excite, MSN, Ubique і Yahoo!).

Кожна така програма мала свій власний протокол та клієнт, тому користувачам довелося запускати декілька клієнтських додатків, якщо вони хотіли використовувати більше одного месенджера. У 2000 році було запущено прикладне програмне забезпечення з відкритим кодом та протокол відкритих стандартів під назвою Jabber. Протокол був стандартизований під назвою Extensible Messaging and Presence Protocol (XMPP). XMPP-сервери можуть діяти як шлюзи для інших протоколів IM, що зменшує необхідність запускати декілька клієнтів.

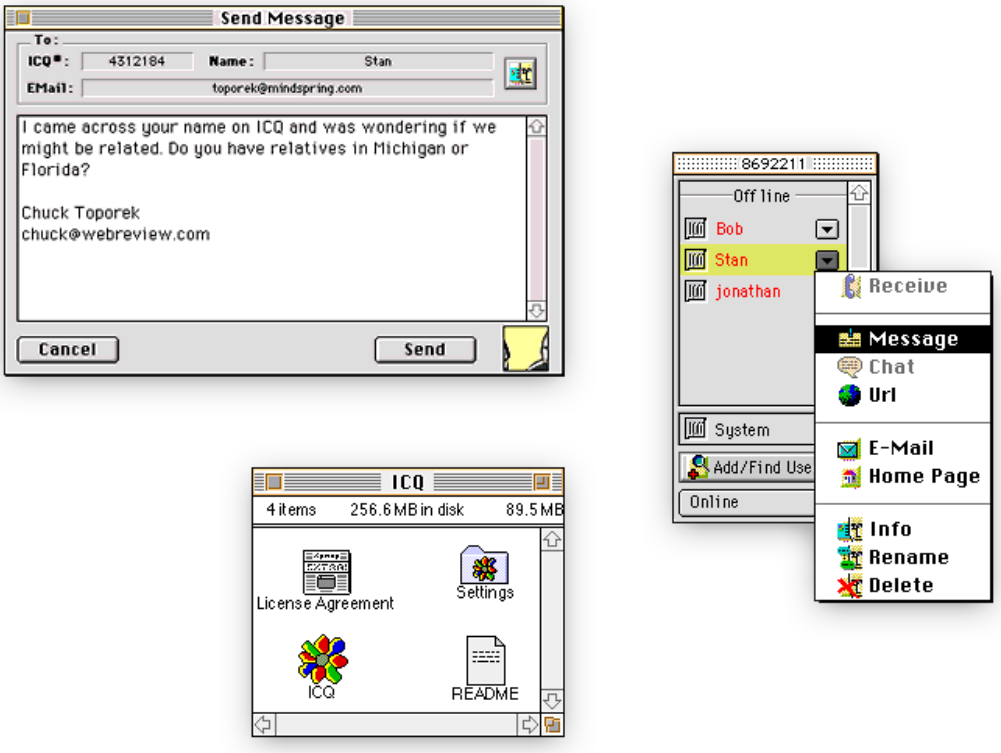


Рисунок 1.1 – Інтерфейс ICQ 1996–1997-х років [2]

Можна бачити, що перший інтерфейс користувача був примітивним, а сама програма надавала мінімальний за теперішніми мірками функціонал. Також у багатьох із перших СМОП було обмеження на максимально допустиму довжину повідомлення, що в купі зі схожим обмеженням на довжину смс-повідомлення призвело до виникнення «сленгу месенджерів». Користувачі іноді використовують Інтернет-сленг, щоб скоротити слова чи фрази для прискорення написання або зменшення кількості символів. Наприклад, широко популярними є такі скорочення

фраз: «lol» (laughing out loud – сміятися голосно), «brb» (be right back – скоро повернуся) і «TTYL» (talk to you later - поговоримо пізніше). Ці та інші скорочення часто застосовуються у неформальних повідомленнях. Дехто навіть використовує їх у повсякденній мові. Загалом вони стали невід'ємною частиною спілкування у мережі, а без їх знання важко буде зрозуміти іноземців у повсякденній переписці.

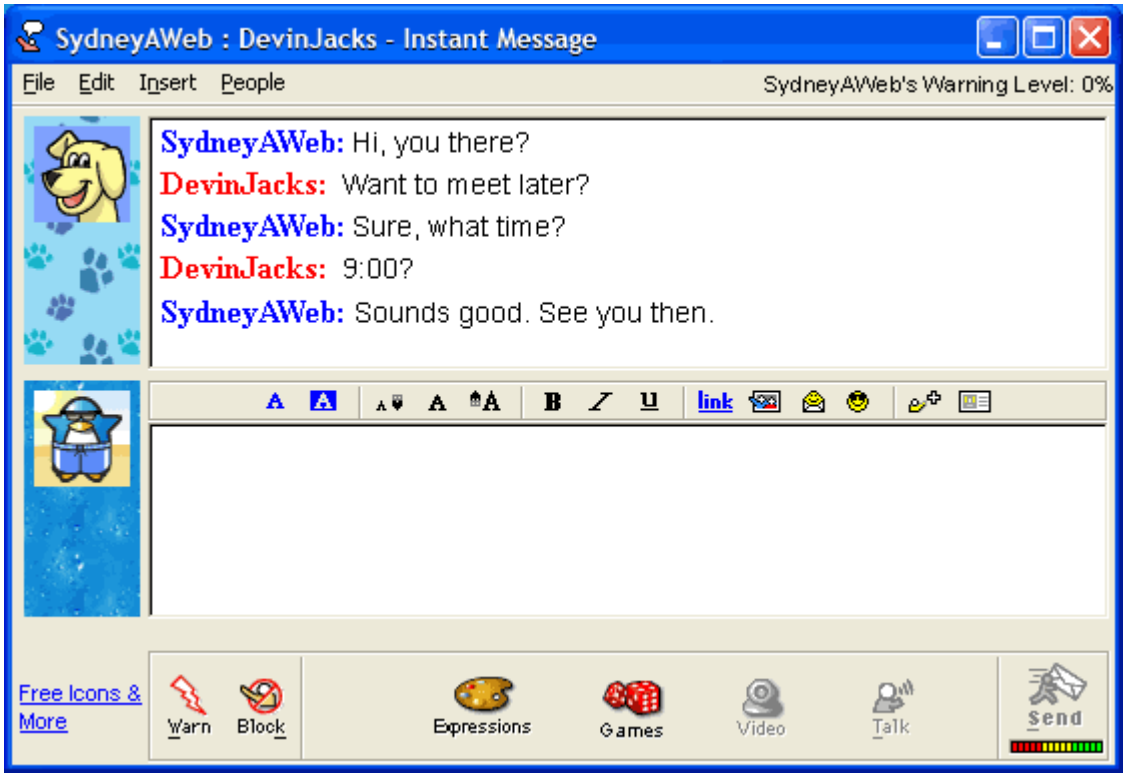


Рисунок 1.2 – Інтерфейс AOL Instant Messenger у 2000 році [3]

AIM, мабуть, є справжнім попередником сучасних сайтів соціальних мереж. Профілі учасників дозволили користувачам написати біографію та ділитися докладними відомостями про себе [3]. Профілі були доступні для пошуку, щоб люди могли переглядати ваш профіль. Це була найбільш інноваційна функція того часу. Першою соціальною мережею, схожою на сучасні була мережа сайту Friendster. Впродовж перших трьох місяців Friendster зміг залучити 3 000 000 користувачів. У той час це означало, що 1 з 126 веб-користувачів були учасниками Friendster. Послідовником Friendster-а став MySpace, який було запущено в серпні 2003-го. Кількість його зареєстрованих користувачів сягає 90 мільйонів.

Facebook був запущений в 2004 році, а його першочерговим наміром було зв'язати студентів американських коледжів. Facebook вперше почав функціонувати у альма-матер Марка Цукерберга – Гарвардському університеті [4]. Спочатку приєднатися до цієї мережі можна було лише у тому випадку, якщо вас запросив користувач Facebook. Така «ексклюзивна» особливість мережі виявилась успішною. Першого місяця зареєструвалося більше половини студентів Гарвардського університету. Через два роки сайт, доступний лише для студмістечок, став відкритим для громадськості. У 2008 році Facebook обігнав MySpace і Friendster у кількості користувачів соціальної мережі. 2017 року кількість користувачів Facebook досягла 2 млрд чоловік у всьому світі, у тому числі 10 млн чоловік в Україні.

1.2 Сучасні месенджери

Розглянемо більш детально сучасні СМОП. Останні роки популярність знову набирають месенджери. Імовірна причина повторної хвилі їх популярності – зміни в області мобільного Інтернету: високі швидкості, нижчі ціни, та широке поширення смартфонів[5].

На графіку (рисунок 1.3) представлено кількість активних користувачів у найпопулярніших месенджерах станом на квітень 2018 року, за даними дослідження сайту statista.com[6]. Як бачимо очевидним лідером є WhatsApp із загальною кількістю користувачів у приблизно 1.5 мільярди. Слідом за ним іде Facebook Messenger із 1.3 мільярдами активних користувачів за місяць.

Очевидно, що всі месенджери у даному списку різні, проте у них є спільні риси, які стали причинами їхнього успіху. Практично у кожного із них зручний, сучасний та зрозумілий інтерфейс користувача. Також кожен із месенджерів підтримує групові чати. Деякі з них підтримують канали, які частково повторюють функціонал чатів, за винятком того, що можливість публікувати повідомлення мають лише користувачі із відповідними правами доступу.

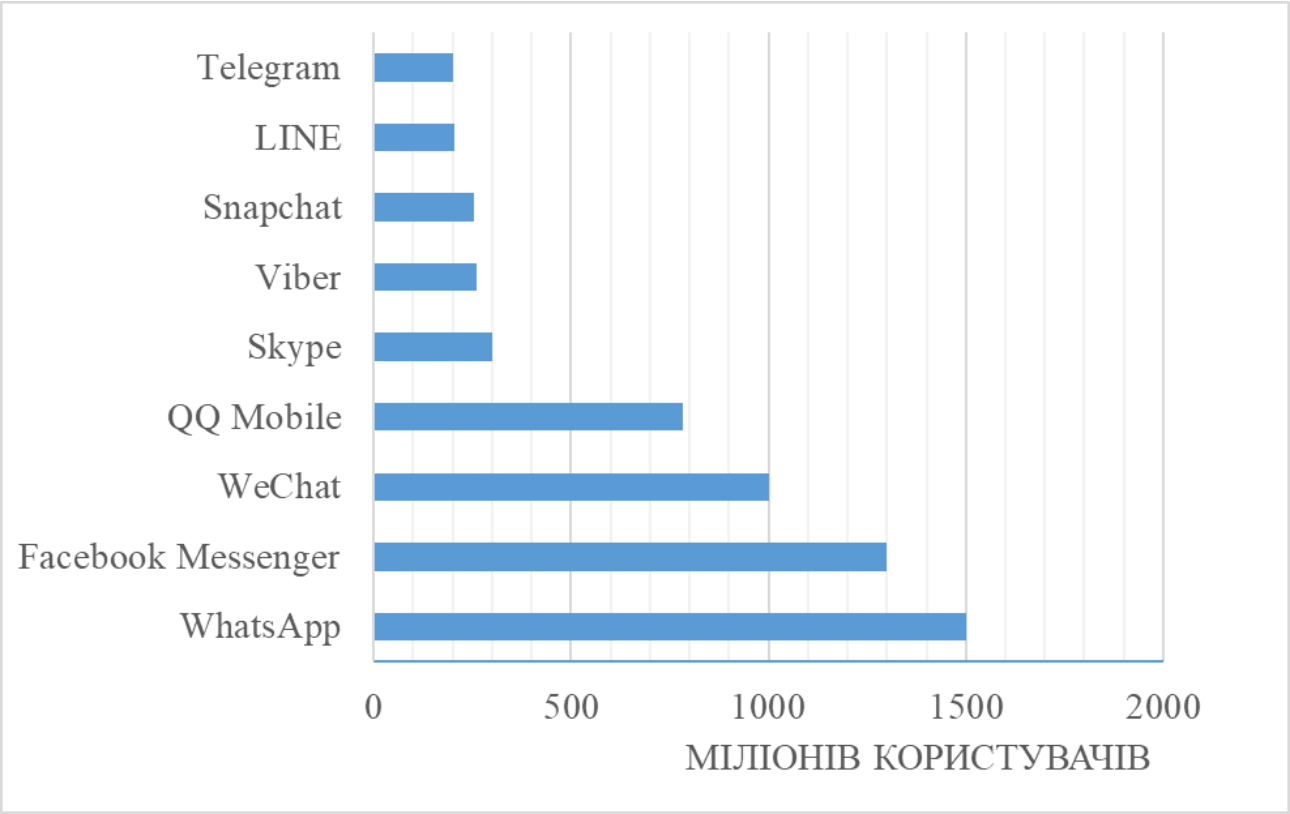


Рисунок 1.3 – Кількість активних користувачів у найпопулярніших месенджерах [6]

Проте для аналізу переваг та недоліків СМОП потрібно також описати певні їх відмінності. Практично кожен із них надає можливості аудіо дзвінків. У багатьох є функція відео дзвінків. Telegram, LINE, Viber, Skype, WeChat та Facebook Messenger підтримують передачу файлів від користувача користувачу. Порівняємо WhatsApp, як найбільшого, за кількістю активних користувачів, та Telegram, як найшвидше зростаючого месенджера.

Далі наведено графік росту кількості активних користувачів месенджера WhatsApp у період із січня 2015 по грудень 2017 (рисунок 1.4). За цей час кількість активних користувачів за місяці подвоїлася із 700 мілионів до 1.5 міліардів. Тобто кожна 5 людина на планеті користується WhatsApp щомісяця.

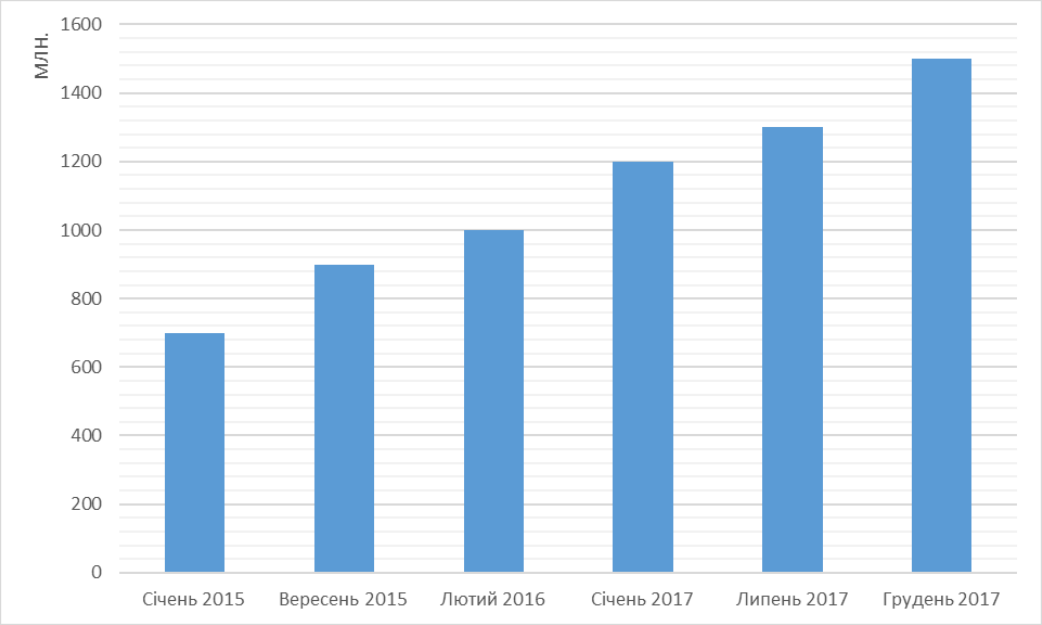


Рисунок 1.4 – Кількість активних користувачів WhatsApp за місяць, млн [7]

Telegram – месенджер, який набуває популярності по всьому світу. В період із лютого 2016 по грудень 2017 кількість активних користувачів зросла на 80% (рисунок 1.5). За той же період у WhatsApp цей показник зріс на 50%. Telegram сміло можна називати найбільш швидко зростаючим месенджером. Станом на квітень 2018 він уже пересік відмітку у 200 мільйонів активних користувачів[8].

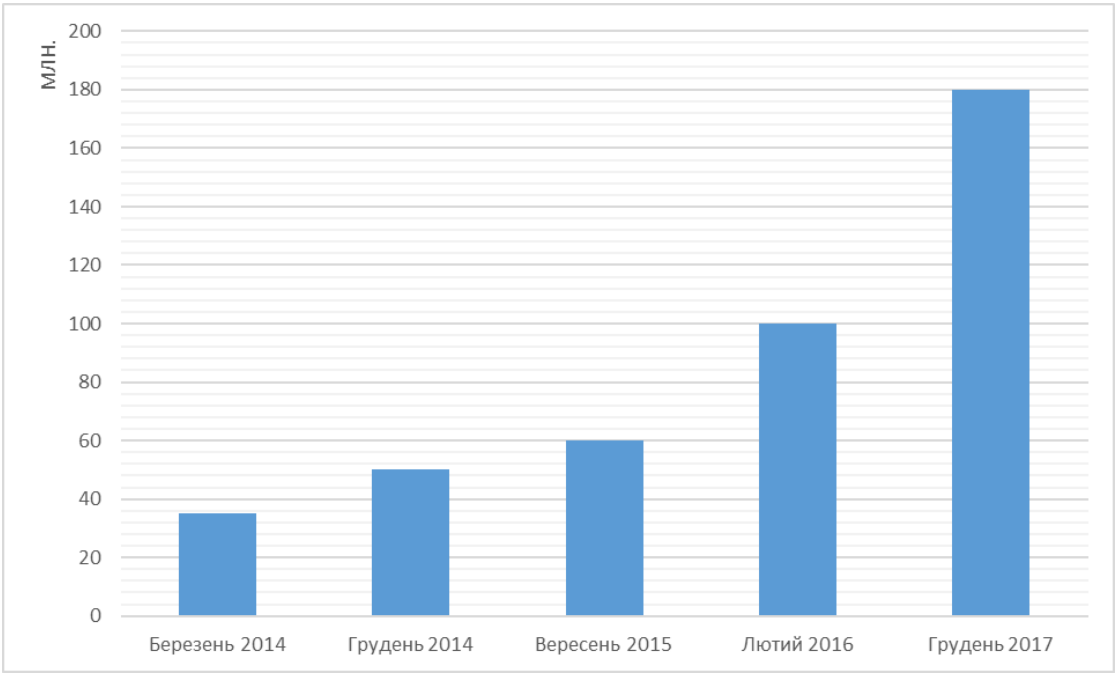


Рисунок 1.5 – Кількість активних користувачів Telegram за місяць, млн [8]

Варто також згадати, що Telegram пропонує можливість секретних чатів, за яких практично не можливо отримати доступ до його вмісту. Також у порівнянні із WhatsApp, Telegram працює швидше та надає можливість передачі будь-яких файлів. Не зручним є також те, що у WhatsApp можливо увійти лише на одному пристрої та, відповідно до правил використання додатку, користувача може були заблоковано при частому перенесенні облікового запису на різні пристрої.

Останні декілька років традиційні телефонні дзвінки стають усе менш популярними у порівнянні із засобами миттєвого обміну повідомленнями чи сервісами онлайн зв'язку. Більше того багато людей та компаній надають перевагу текстовим чи мультимедійним повідомленням у порівнянні із дзвінками.

Як показав огляд сучасних СМОП, вони є важливою частиною сучасного спілкування у мережі. Користувачі надають перевагу зручному, дешевому та швидкому спілкуванню, тому обирають месенджери. Звісно існує велика кількість додатків цього типу, проте, саме Telegram має найбільші темпи росту кількості активних користувачів, а тому й перспективи росту аудиторії для розроблюваного сервісу. Отже, питання розробки та наукового супроводження СМОП, на поточний час, є актуальним.

2 БОТИ ЯК ВАЖЛИВА ЧАСТИНА СПІЛКУВАННЯ У МЕРЕЖІ

Усі, наведені у попередньому розділі СМОП, так чи інакше, підтримують інтеграцію із програмами-ботами. Snapchat та QQ Mobile підтримують так звану сіру інтеграцію – офіційно вони не підтримують використання ботів, та не регулюють використання сервісів програмами під виглядом користувачів. Усі інші підтримують інтеграцію ботів на рівні платформи, що означає підтримку їх створення за допомогою надання прикладного програмного інтерфейсу (API) та докладної документації платформи.

Важливо розуміти, що існує велика кількість різноманітних ботів, які так чи інакше, беруть участь у життях людей уже зараз. При відвідуванні веб-сайту можна зустріти бота-помічника, який зможе відповісти на прості питання про товари. При виникненні проблем цей бот зможе направити запит до працівника підтримки сайту на основі наданих користувачем даних. Також у багатьох магазинів існують боти для розсилки інформації про товари та для швидкого їх замовлення без покидання месенджерів.

2.1 Класифікація ботів для СМОП

Класифікувати ботів можна за декількома параметрами. За способом отримання інформації від користувача боти поділяються на:

а) Текстові – які отримують інформацію від користувача у текстовій формі та обробляють її для виділення команд;

б) Голосові – які перетворюють людське мовлення на текст, а вже тоді його аналізують [9].

Текстові боти є простішими у розробці та швидшими, за рахунок того, що немає затримки обробки інформації. Проте дана затримка зменшується з кожним роком за допомогою нових алгоритмів обробки голосової інформації та збільшення потужностей сучасних пристроїв. Голосові помічники є зручнішими у багатьох випадках, оскільки вони можуть бути повноцінними співрозмовниками. Крім того,

вони можуть сприймати та розуміти людську мову, вони, зазвичай, здатні синтезувати відповіді у вигляді аудіо. Це дозволяє вести діалог із такою програмою.

Одним з перших віртуальних співрозмовників була програма Еліза, створена в 1966 році Джозефом Вейзенбаумом. Еліза пародіювала мовну поведінка психотерапевта, реалізуючи техніку активного слухання, перепитуючи користувача і використовуючи фрази типу «Будь ласка, продовжуйте».

Для найбільш розвинених програм-співрозмовників існує окремий термін – «віртуальний помічник». Вони здатні, за голосом, впізнати хто до них звертається та врахувати це при відповіді. Наприклад, при звертанні дитини, помічник видаватиме тільки доречний для дітей контент. Також ведеться активна розробка засобів для аналізу емоційного забарвлення голосу користувача. Тобто, на основі тону голосу та підбору слів користувачем, бот може визначити яку відповідь краще надати [10]. Вважається, що ідеальна програма-співрозмовник повинна пройти тест Тьюринга. Цей тест, призначений для перевірки здатності машини демонструвати розумну, людську поведінку. Щоб пройти тест, відповіді програми не повинні відрізнятися від відповідей людини впродовж п'ятихвилинного тесту.

За призначенням чат-ботів поділяють на:

а) ЧБ для розмов на широкий спектр тем;

б) Орієнтованих на певну мету.

Перші призначені для діалогу із користувачем на абстрактні теми та без певної чіткої мети. Другі орієнтовані на вирішення буденних проблем засобами «природньої» мови [11]. Боти орієнтовані на допомогу користувачу у досягненні певної мети (регулярне отримання корисної інформації, встановлення нагадувань, тощо) є найбільш поширеними та саме про них йтиметься далі у дисертації.

Як і при роботі із програмою, при використанні бота важливий інтерфейс взаємодії із користувачем. Інтерфейс власне месенджера чітко визначений, проте інтерфейс бота це поняття, в певному сенсі, більш розмите. Він включає не тільки спосіб введення інформації, але й методи взаємодії бота із цією інформацією.

До взаємодії бота із інформацією можна віднести підтримку команд, можливості виокремлення команд із повідомлення користувача та можливість розуміти контекст діалогу.

2.2 Примітивні боти

Підтримка команд – це те, що відрізняє бота від звичайного користувача. Примітивні боти обмежуються цим функціоналом. У таких ботів є певний обмежений набір доступних команд, які вони здатні обробити та виконати пов’язану із ними дію.

Часто для зручності користувача ці команди подаються візуально у інтерфейсі месенджера і для їх відправки достатньо просто натиснути по ній на екрані (рисунок 2.1).

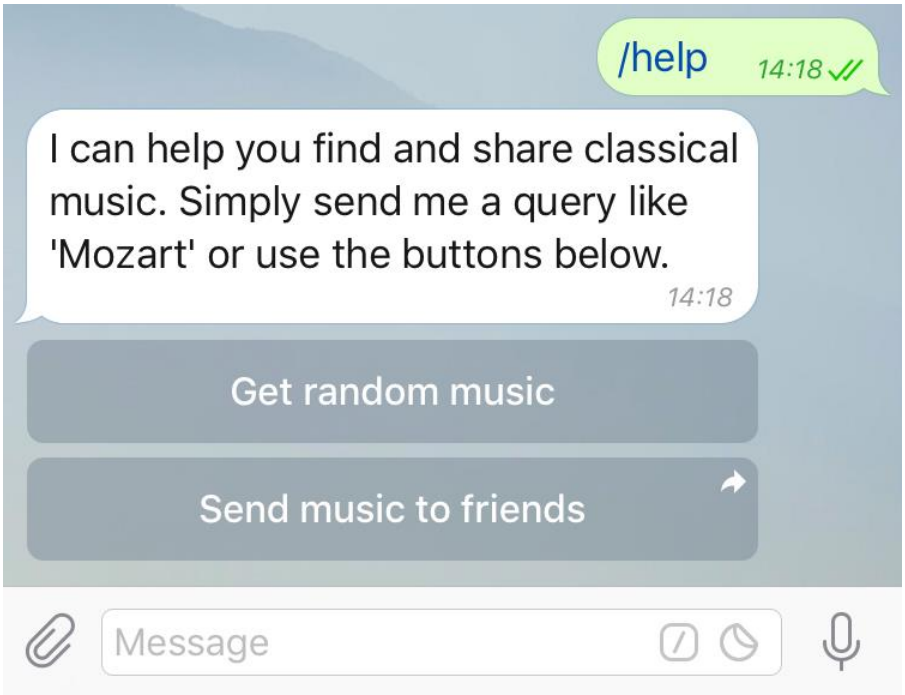


Рисунок 2.1 – Приклад команд-кнопок у інтерфейсі бота для Telegram [12]

Загальноприйнятою концепцією є написання фіксованих команд боту використовуючи символ «/» як префікс. Наприклад «/help» - команда реалізована для багатьох ботів, яка зазвичай виводить список доступних команд чи іншу необхідну користувачу інформацію для початку роботи із ботом [13]. На рисунку 2.2 наведено

приклад відповіді на таку команду ботом BotFather, який дозволяє створити бота для месенджера Telegram. У команд такого типу не допускається наявності пробілу між словами в команді. Наприклад, команда «/set name» вважатиметься невалідною.

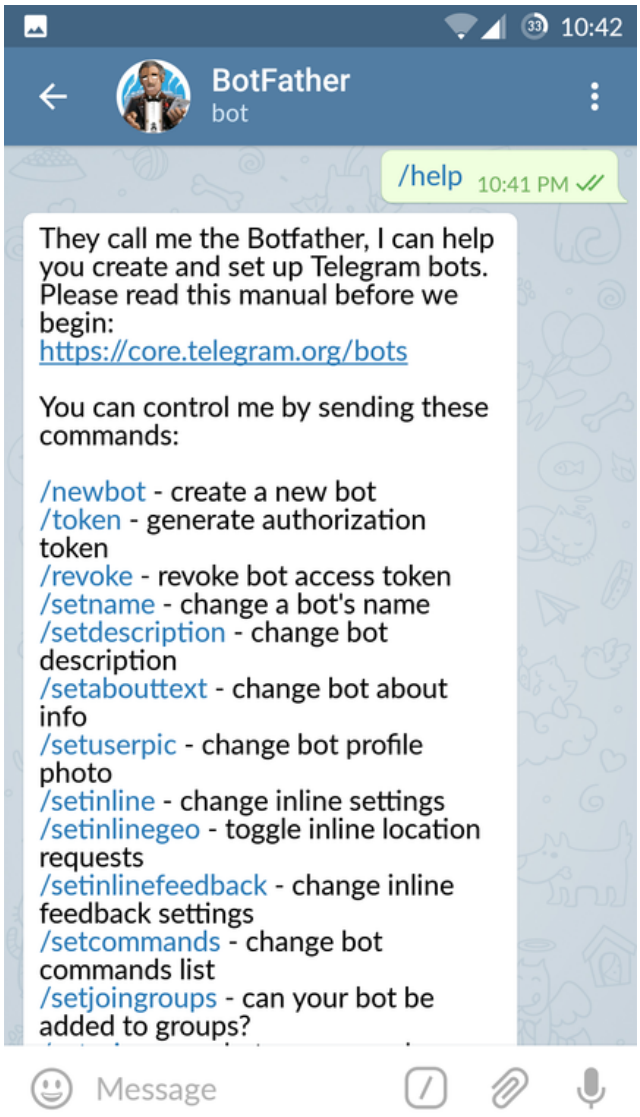


Рисунок 2.2 – Результат обробки команди «/help» ботом BotFather [13]

Також до таких команд можливо додавати певний текст. До прикладу, можна реалізувати таку команду: *«/встановибудильник на 10 ранку»*. Таких ботів найпростіше розробляти, а введення нових функцій потребує просто внесення нової команди в програму.

2.3 ЧБ, які здатні виокремити команду з повідомлення

Виокремлення команд із повідомлення («розуміння») користувача значно підвищує привабливість бота, а тому стає все популярнішою функцією. До прикладу, при наявності такого функціоналу бот розумітиме різноманітні варіації команди, наприклад: *«Чи не міг би ти установити будильник на 10:00 26-го червня» чи «Установи будильник на завтра на 10:00»,* що є дуже схожим на «природню» розмову. За таких умов необхідне використання засобів обробки природної мови (NLP) [14]. Людська мова часто є неточною і в залежності від предмета та конкретної мови, що розглядається, одне слово може означати 500 різних речей. Наприклад, англійське слово «run» має 645 різних значень.

Засоби обробки природного мовлення визначають значущі фрагменти інформації, наданої користувачем і на їх основі формують правильну відповідь. Ці засоби використовують розпізнавання названого об'єкта (англ. Named Entry Recognition – NER) та токенізацію, щоб точно визначити те, що хоче користувач.

Розпізнавання названого об'єкта - це тип вилучення інформації, який призначений для ідентифікації та класифікації названих об'єктів у визначених категоріях, включаючи, але не обмежуючись такими [15]:

а) Компанії та організації;

б) Місцеположення;

в) Імена людей чи назви продуктів;

г) Позначення часу.

Таким чином, система розпізнавання буде знати, що Джейн – це ім’я, 2013 рік – це 2013-й рік в григоріанському календарі, а Сан-Франциско – це місто.

Щоб ЧБ зрозумів речення, йому спочатку потрібно розділити його і перетворити на серію «токенів». Ці токени можуть мати будь-яку форму – від слова до числа або будь-якого знаку пунктуації. Це також можна назвати сегментацією речення. Бот потім буде класифікувати всі токени за контекстом.

Десь на межі між цим та попереднім видом ботів існують боти, що виділяють ключові слова із фраз користувачів. В таких ботів є заготовані фрази для відповіді на

запитання із певним набором ключових слів. Наприклад, слово «акція» для бота на сайті роздрібної компанії призведе до використання попередньо прописаної для цього слова відповіді: «Сьогодні взуття за акційною ціною!». Але якщо такого бота запитали щось більш складне або серію запитань – він або попросить вас уточнити запитання, або надасть вам абсолютно непідходящу відповідь.

2.4 Боти які здатні розуміти контекст діалогу

Наступним витком еволюції чат-ботів є можливість розуміти загальний контекст діалогу. Для прикладу можливості розуміти контекст діалогу наведемо приблизно можливий діалог (Л – людина, Б- бот):

Л – Додай банани у список покупок.

Б – Додано.

Певна перерва в діалозі, або розмова на іншу тему.

Л – Також потрібно купити упаковку борошна.

Б – Борошно додано у список покупок.

Результатом стане список покупок, що міститиме і банани, і упаковку борошна. Слід також звернути увагу на різні варіанти відповіді від бота. Це ще більше наближає штучний діалог до діалогу між людьми. У даному випадку, для гнучкості розробки бота та для забезпечення можливості зрозуміти зв’язаність повідомлень, необхідним є застосування нейронних мереж та машинного навчання.

Схожим застосуванням є використання нейронних мереж для персоналізації та вдосконалення виводу інформації користувачеві. Наприклад, бот проаналізував недавнє спілкування та прийшов до висновку що користувач надає перевагу процесорам марки Intel, тому на запит *«Я хотів би купити процесор»* він відповість інформацією про продукцію даної марки.

Сучасні боти також можуть інтегруватися із сторонніми сервісами. Наприклад, якщо сказати голосовому помічнику, від практично будь якої компанії, щоб він включив музику – він включить її в улюбленій програмі для відтворення музики користувача, якщо, звісно, ця програма підтримує таку інтеграцію. На рисунку 2.3

наведено приклад використання Google Assistant (віртуального помічника від компанії Google) на телефоні для відтворення відео на телевізорі та керування відтворенням у режимі діалогу із ботом.

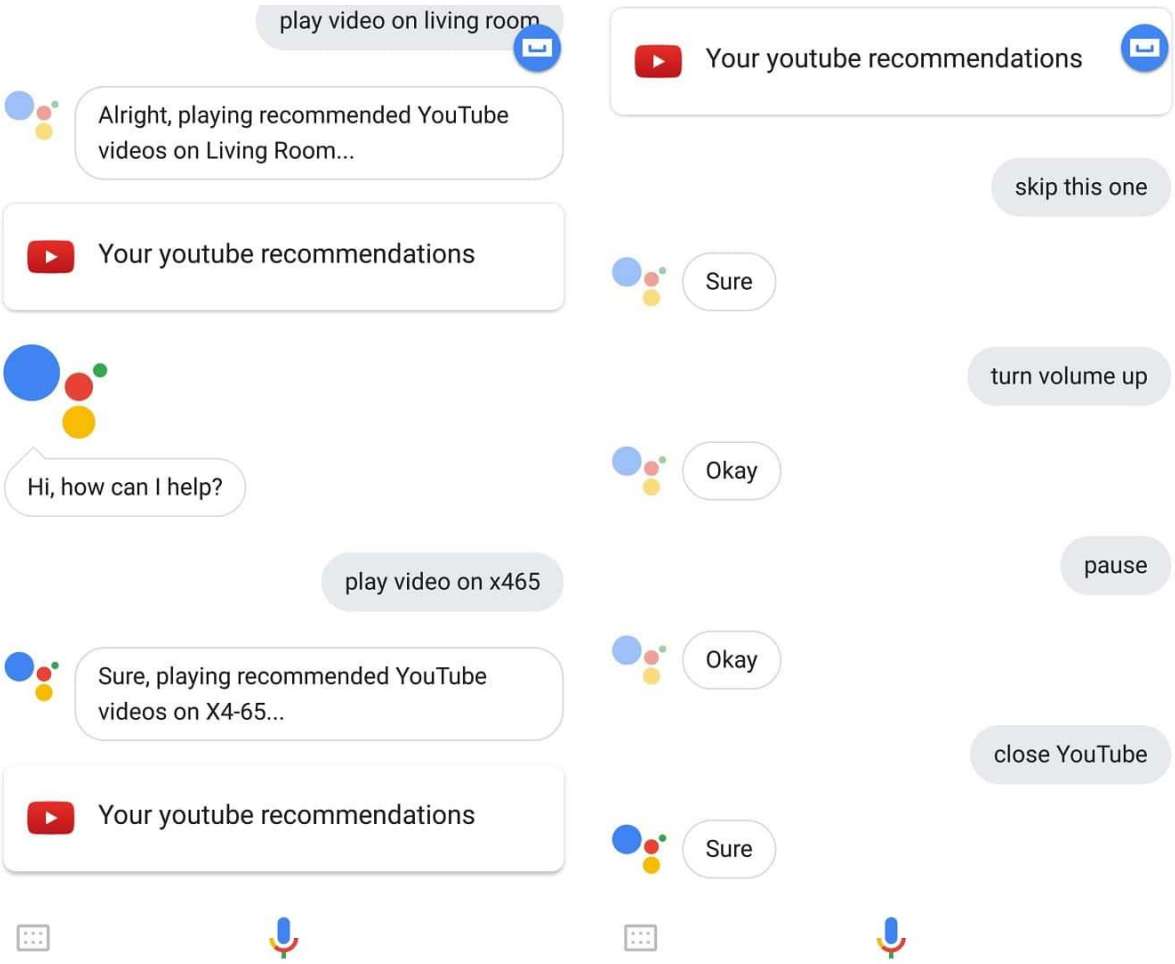


Рисунок 2.3 – Використання Google Assistant на телефоні для відтворення відео на телевізорі [16]

2.5 Приклади застосування ботів

2.5.1 Застосування ботів звичайним користувачем

Чат боти найвідоміші тим, що використовуються як агенти служби підтримки клієнтів. Вони здатні відповісти на більшість запитань клієнта без необхідності втручання людини. Проте крім цього існує безліч інших корисних застосувань для таких програм. Існують боти, що дозволяють забронювати готельні номери, скласти

людині компанію, організовувати зустріч для групи людей, відправити гроші користувачам, тощо.

Один з найновіших ЧБ, який сколихнув увесь світ – це безкоштовний бот-юрист. Незважаючи на те, що він не скоро замінить справжнього адвоката, він може допомогти вам створити справи для оскарження штрафів за неправильне паркування, отримати компенсацію за неочікувані витрати при подорожах або надасть відповіді на різноманітні юридичні питання [17].

Загалом ЧБ можна запрограмувати на практично будь-що. Деякі навіть здатні розуміти емодзі та картинки, реагуючи на них схоже на людину. Прикладом може бути той факт, що коли бот, розроблений компанією Microsoft, Xiaoice отримав фотографію людини, яка з ним розмовляла із підвернутою ногою, він запитав наскільки було боляче при травмі.

Прикладом використання, відомим сайтом, функціоналу бота є - AutoTLDR бот від сайту Reddit. Більшість людей знають Reddit як «обличчя Інтернету». Це спільнота, де можна поділитися практично всім на форумах, які називаються субреддіти (subreddits). Однією із багатьох функцій, які Reddit пропонує своїм користувачам є AutoTLDR (Too Long, Do not Read) бот. Він призначений для тих, хто хоче ознайомитися з новинами, проте не має часу читати всю статтю. За короткий проміжок часу бот створює резюме статті довжиною у 450-700 символів, розміщуючи його на форумах разом із посиланням на повноцінну статтю [18].

Іншим прикладом використання ботів є «DR. A. I. ♥» - це чат бот «лікар». Він може забезпечити базовий діагноз для багатьох поширених захворювань, таких як бронхіт, розтягнута щиколотка, або лихоманка, задаючи важливі і осмислені питання. «DR. A.I. ♥» набагато відрізняється від простого використання пошукової системи для пошуку причини певних симптомів, оскільки він діє так само, як і звичайний лікар, запитуючи навідні запитання та надаючи рекомендації щодо подальших кроків. Якщо вам потрібне заключення фахівця, «DR. A.I. ♥» також може надіслати такий запити реальному лікарю, щоб отримати схвалення. При цьому користувачу не потрібно виходити з дому. Звичайно, «DR. A.I. ♥» не повинен використовуватися в надзвичайних ситуаціях і не є повною заміною справжнього лікаря [17].

Mezi (рисунок 2.4) – ще один корисний бот, який дозволяє легко спланувати подорож. З його допомогою можна забронювати квитки, готелі та навіть вечерю у ресторані. Скажіть ЧБ, коли і куди ви хочете поїхати і звідки ви хочете полетіти і він підбере та забронює рейс та готель замість вас. Цей бот усуває потребу у пролистуванні десятка веб-сайтів для бронювання авіа перельотів та готелів, а надає зручний інтерфейс для їх пошуку [17].

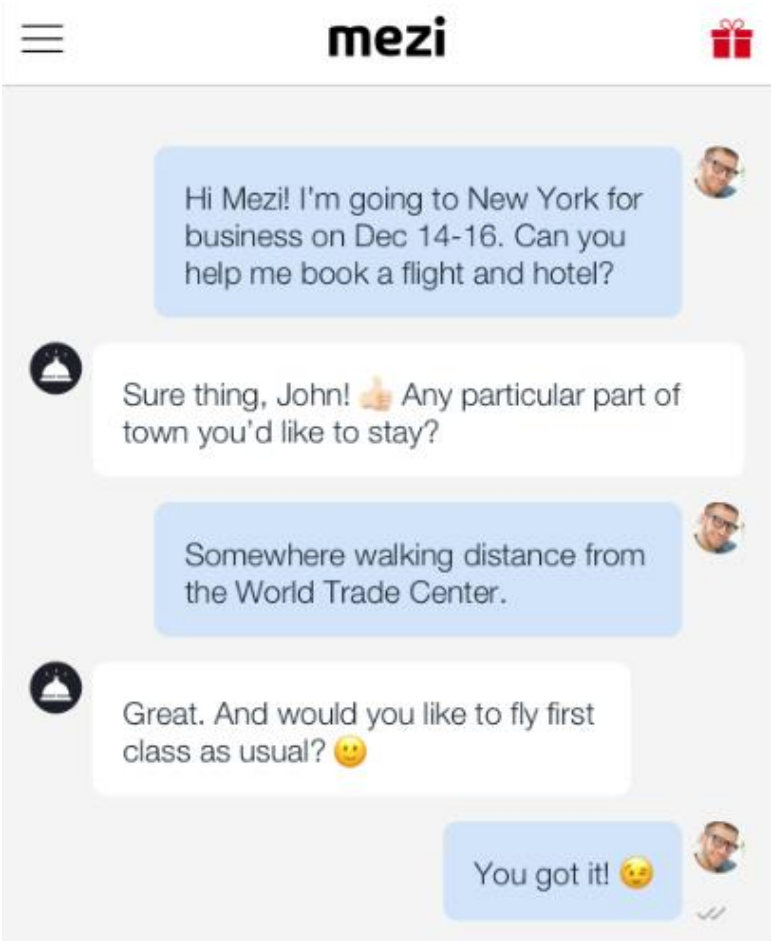


Рисунок 2.4 – Інтерфейс бота Mezi

Надзвичайно популярними є боти для доставки їжі. Вони допомагають вам замовити їжу та її доставку, без необхідності дзвонити до ресторану. Все більше і більше компаній використовують такого помічника, від мереж доставки піци до крафтових броварень (рисунок 2.5). Для замовлення потрібно просто вказати ЧБ необхідні товари, підтвердити свою інформацію та здійснити оплату. Бот проробить усі необхідні функції для того щоб користувач отримав їжу у визначений час.

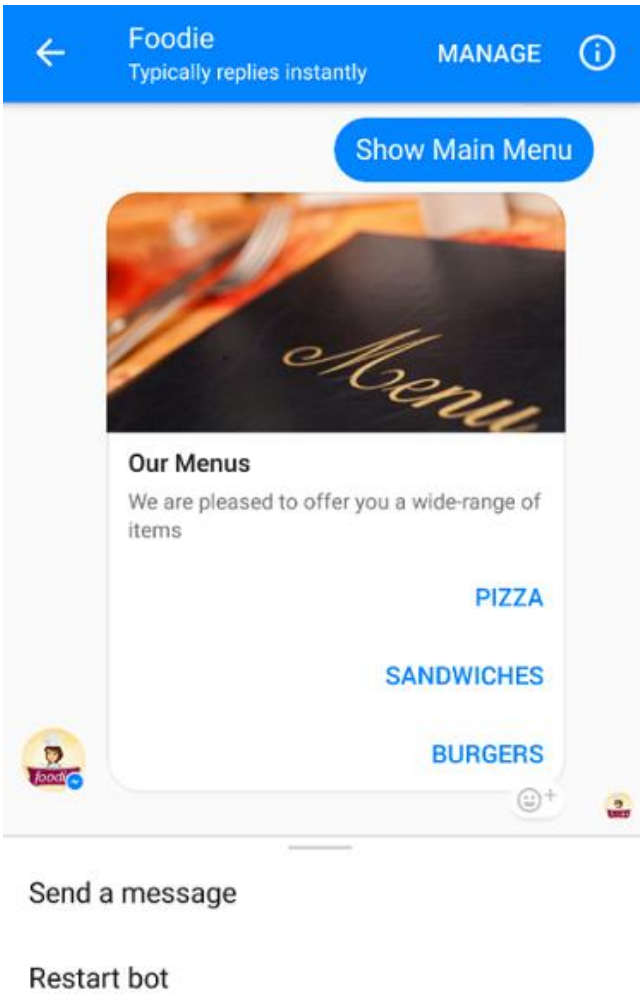


Рисунок 2.5 – Інтерфейс бота для доставки їжі Foodie у Facebook Messenger [19]

Ще одним дуже популярним типом ботів є новинні агрегатори. Подібно до RSS каналу, бот-агрегатор новин може тримати вас в курсі всіх новин, про які ви хочете дізнатися. Компанії, як великі, так і малі, впроваджують власних агрегаторних ботів, які надають різноманітну інформацію, від світових новин до дуже специфічного списку того, що користувач вважає необхідною інформацією. Відмінним прикладом може служити Flash Briefing від Amazon, за якого Alexa (голосовий помічник від компанії Amazon) може прочитати короткий виклад деяких популярних новин.

Існують також і більш екстравагантні, за своїм призначенням ЧБ. Наприклад, бот для тих, кому не спиться уночі. Insomnobot (рисунок 2.6) був створений компанією виробником матраців Casper. Найактивніше цей бот використовується з 11 вечора і до 5 ранку. Insomnobot підтримає розмову про улюблені закуски, телепередачі або плани на вихідні.

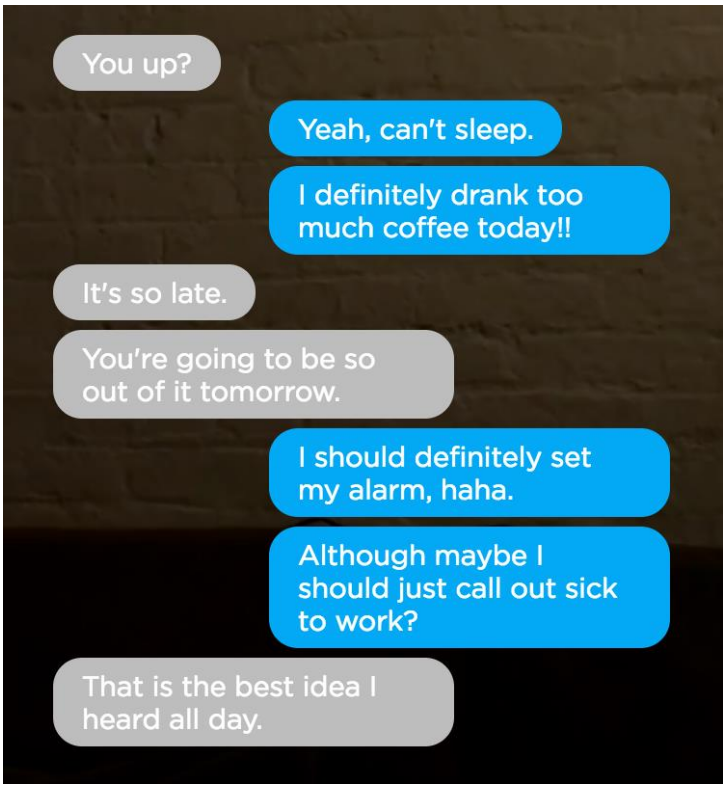


Рисунок 2.6 – Insomnobot підтримує розмову із користувачем [20]

2.5.2 Застосування ЧБ та СМОП для потреб бізнесу

Не важливо за яким принципом функціонує ваш бізнес – за B2B або B2C, ЧБ здатні покращити обслуговування клієнтів, автоматизувати повторювані завдання та звільнити значну кількість часу. Далі наведено лише деякі переваги, які надає ЧБ для бізнесу.

У випадку B2B моделі бот може покращити декілька сфер. У сфері внутрішніх комунікацій ЧБ можна використовувати для покращення зв'язку між користувачами, відділами та локаціями завдяки можливості оголошувати новини, зміни або оновлення автоматизованим ботом.

У роботі з асистентами, зосереджуючись на більшій кількості адміністративних завдань, боти-помічники допомагають вам документувати ваш розпорядок, тримати вас у курсі всіх ваших завдань та навіть нагадувати вам про зустрічі чи подію. Віртуальні помічники частково можуть замінити секретарів для деяких керівників.

При використанні ЧБ як інтерфейсу доступу до бази знань, він дозволяє отримати швидкий, легкий доступ до документів чи інформації без необхідності шукати її в величезному сховищі.

У сфері статистики боти корисні тим, що можуть автоматично формувати статистичні дані у зручний користувачу формат. Наприклад InsightBot може відслідковувати ціни на акції, перегляди сторінок веб-сайту компанії, або кількість контактів, які було створено за попередній день.

Звичайно, не всі переваги суто внутрішні. Є багато причин для того, щоб B2B розглянув використання зовнішньо-направленого бота разом із внутрішньо-направленим.

Наприклад, можна використовувати ЧБ для відповіді на запитання потенційного чи існуючого клієнта, який можливо не готовий (або не хоче) до прямих переговорів з справжнім продавцем. Він може скористатися чат ботом для отримання відповідей на питання про товари компанії, або для отримання певної поради;

Також, ЧБ може надати актуальну інформацію про те чим цікавились користувачі: такий бот веде журнал комунікацій, що відбуваються між ним та клієнтами. Це дає бізнесу безпосереднє уявлення про те, що клієнти шукають, про що вони запитують бота та про те, що потрібно покращити у сайті чи продукті [17].

Бізнес, який використовує B2C модель має деякі особливо корисні переваги при використанні чат бота, при тому, що він все ще здатний отримувати ті ж переваги що були описані для B2B моделі.

Для B2C неймовірно важлива миттєва доступність, а тому те, що ЧБ працює в режимі онлайн цілодобово є суттєвою перевагою. Не всі користувачі використовують можливості покупки товару на сайтах лише з 9 ранку до 5 вечора, тому можливість отримати допомогу, за необхідності, у будь який час доби є суттєвою перевагою [17].

Покращення продаж, формування пропозицій на основі попередніх покупок і навіть здійснення платежів – це все можливі переваги використання бота для електронної комерції. Іншим прикладом такого бота може бути помічник, що допомагатиме при покупці, надаватиме деталі замовлення, інформацію про доставку, а іноді навіть допомагатиме при необхідності повернути товар, якщо той не підійшов.

ЧБ були практично задумані, як агенти для обслуговування клієнтів. Замість того, щоб утримувати центр обробки викликів, який буде завалений дзвінком про все, що тільки можна, доцільно використати чат бота, який зустрічатиме клієнтів та даватиме відповіді на основні запитання. Якщо чат бот не здатний відповісти на запитання або не володіє інформацією яка необхідна користувачу, він може з’єднати того із живим агентом підтримки. Очевидно, що таке використання значно скоротило б необхідність у персоналі підтримки користувачів [17].

Деякі банки використовують чат ботів, щоб допомогти клієнтам при онлайн-банкінгу, деколи навіть без необхідності заходити на веб-сайту банку. Замість цього ви просто спілкуєтесь з ЧБ через СМОП, щоб отримати від нього необхідну інформацію.

2.6 Огляд ЧБ з відкритим кодом

Для опису загальних принципів побудови ЧБ для СМОП необхідно розглянути декілька проектів із відкритим кодом. Після цього можна буде виділити фундаментальні особливості створення ботів та їх архітектури.

У додатку А наведено взаємозв'язок елементів при використанні бота для СМОП. Основними взаємодіючими елементами є користувач, платформа Телеграм та сервер Bot API та сервер бота. При надсиланні повідомлення користувачем воно потрапляє на сервер Телеграм і там зберігається доки не буде оброблене ботом. При потраплянні повідомлення на сервер телеграм відбувається подія, на обробку якої бот підписується за допомогою веб-хуків. Веб-хук у розробці - це спосіб доповнення або зміни поведінки веб-сторінки чи веб-додатка, за допомогою зворотних викликів [21]. Повідомлення тоді потрапляє на сервер, на якому розміщено ЧБ. Бот обробляє повідомлення, можливо звертається до сторонніх API для отримання додаткової інформації та формує відповідь користувачу. Сформована відповідь передається на сервер Телеграм і зберігається там поки користувач не прочитає її.

2.6.1 Приклад архітектури серверної частини бота

Розглянемо узагальнений приклад серверної архітектури бота на рисунку 2.7.

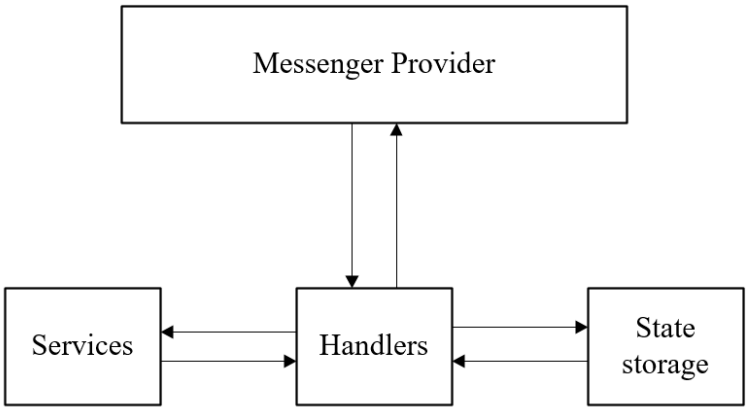


Рисунок 2.7 – Узагальнений приклад серверної архітектури бота [22]

У даному випадку бот складається із чотирьох основних частин:

а) Провайдер месенджера (Messenger Provider);

б) Обробники команд (Handlers);

в) Сервіси (Services);

г) Контейнер стану (State storage).

Провайдер месенджера обгортає Telegram Bot API і забезпечує універсальний інтерфейс для прийому, обробки та відправки повідомлень. Провайдер месенджера визначає, що хоче користувач, і спрямовує його до правильного обробника. Мета цієї частини забезпечити шар абстракції над API месенджерів. Тобто, якщо необхідно буде підтримувати більше СМОП або перейти на використання іншого месенджера то єдиний код, який потрібно буде змінити це ця частина. Розглянемо приклад бота «JonathanZWhite/bot-server». На рисунку 2.8 зображено конструктор та функцію listen класу Messenger, який являє собою провайдер месенджера.

```
class Messenger {
  constructor() {
    this.bot = new TelegramBot('Your token here', { polling: true });
  }

  listen() {
    this.bot.on('text', this.handleText.bind(this))
  }

  ...
}
```

Рисунок 2.8 – Частина класу Messenger [22]

У конструкторі класу Messenger ми створюємо екземпляр бібліотеки Bot API. У цьому випадку використовується API Telegram, але його можна легко замінити. Далі ми маємо функцію listen, яка отримує вхідні ключі чату та передає повідомлення до функції «handleText», яку наведено на рисунку 2.9.

```
class Messenger {
  ...

  handleText(msg) {
    const message = new Message(Message.mapMessage(msg))
    const text = message.text

    if (inputParser.isAskingForGenreList(text))
      return handlers.music.getGenreList(message, this.bot)

    if (inputParser.isAskingForNumberOfRec(text, store.getState(message.from).command))
      return handlers.music.getNumOfRec(message, this.bot)

    if (inputParser.isAskingForRecommendation(text, store.getState(message.from).command))
      return handlers.music.getRecommendation(message, this.bot)

    // default
    return handlers.casual.getHelp(message, this.bot)
  }
}
```

Рисунок 2.9 – Функція «handleText» класу Messenger [22]

Всередині функції «handleText» ми передаємо повідомлення та останню інформацію про стан в із контейнера стану у клас InputParser (рисунок 2.10), яка визначає, що саме хоче користувач, і допомагає відправити команду коректному обробнику.

```
class InputParser {
  isAskingForGenreList(text) {
    const pattern = /music|recommendation/i

    return text.match(pattern)
  }

  isAskingForNumberOfRec(text, prevCommand) {
    return prevCommand === commands.GET_GENRE_LIST
  }

  isAskingForRecommendation(text, prevCommand) {
    return prevCommand === commands.SET_NUMBER_OF_REC
  }
}
```

Рисунок 2.10 – Клас InputParser [22]

InputParser використовує або простий регулярний вираз, або переглядає стан, щоб надати відповідь класу Messenger [22]. InputParser також дозволяє додавати інтеграцію до платформ обробки природної мови або інших засобів нейронних мереж.

Роль обробників полягає в обробці повідомлення, переданого від провайдера месенджера, делегування завдань сервісам, оновлення контейнеру стану, якщо це необхідно, та формування повідомлень для відправлення їх назад користувачеві. На рисунку 2.11 зображено код обробника, що відповідає за роботу із музикою.

Цей клас відповідає за всі дії та команди, пов'язані з отриманням рекомендацій сервісу Spotify для користувача. У цьому прикладі функція «getGenreList» приймає повідомлення і екземпляр бота, що передаються їй із класу Messenger. Повідомлення

– це об'єкт, який містить ідентифікатор користувача та текст, який вони надіслали. Бот – це те як ми надсилаємо повідомлення користувачеві [22].

```
class Music {
  getGenreList(message, bot) {
    // clears store for new command tree
    store.clearState(message.from)
    store.update(message.from, { command: commands.GET_GENRE_LIST })

    bot.sendMessage(message.from, 'Choose a genre you would like a recommendation for 🎵', {
      reply_markup: {
        keyboard: genresJSON.genres,
        one_time_keyboard: true
      }
    })
  }

  ...
}
```

Рисунок 2.11 – Клас Music [22]

Розробники, як правило, оновлюють контейнер стану. У наведеному прикладі ми повідомляємо його про те, де ми знаходимося в дереві розмови. Після оновлення контейнеру стану функція «getGenreList» надсилає користувачеві список жанрів Spotify для вибору.

Є моменти, коли потрібна складніша бізнес-логіка, щоб з'ясувати, що відправити назад користувачеві. У цих ситуаціях необхідно передавати обробку сервісам (рисунок 2.11).

Всередині функції «getRecommendations», як і в «getGenreList», спочатку оновлюється контейнер стану. У цьому випадку також оновлюємо ще одну частину збереженого стану, щоб система пам'ятала, скільки музичних рекомендацій хоче користувач.

Оскільки потрібно зробити виклик API Spotify, щоб отримати рекомендації, викликаємо функцію «getRecommendation» з класу MusicService, який є сервісом для

отримання даної інформації. Він повертає список рекомендацій пісень, які ми можемо надіслати користувачеві.

```
class Music {
  ...

  getRecommendation(message, bot) {
    store.update(message.from, {
      command: commands.GET_RECOMMENDATION,
      spotify: {
        numberOfRecs: message.text
      }
    })

    const state = store.getState(message.from)
    const selectedGenre = state.spotify.genre
    const numberOfRecs = state.spotify.numberOfRecs

    return MusicService.getRecommendation(selectedGenre, numberOfRecs)
      .then(sendRecommendation)

    function sendRecommendation(resp) {
      if (!resp.length) return bot.sendMessage(message.from, 'Sorry, looks like I don\'t have anything

      bot.sendMessage(message.from, 'Why not give this a listen 💃')

      resp.forEach((song) => {
        bot.sendMessage(message.from, song)
      })

      store.clearState()
    }
  }

  ...
}
```

Рисунок 2.12 – Класу Music: використання MusicService [22]

MusicService є прикладом того, які функції повинні виконувати сервіси. Замість того, щоб поміщати цю логіку в обробники, ми використовуємо сервіс, який виконує складну та відносно довготривалу логіку. У сервіси слід поміщати запити до сторонніх API, обробку даних та форматування відповідей для обробників.

Проведення чіткого розмежування відповідальності між обробниками та сервісами спрощує тестування та масштабування коду.

У фрагменті коду на рисунку 2.13 MusicService робить запит до API Spotify, формує відповідь за його результатами і передає її назад обробнику.

```
(function(module) {

  function getRecommendation(selectedGenre, limit) {
    ...

    return Promise.resolve()
      .then(getSpotifyAccessToken)
      .then(getRecommendation)

    function getRecommendation() {
      return spotify.getRecommendations({
        seed_genres: selectedGenre,
        limit: limit
      })
      .then((resp) => {
        return _parseForTrackList(resp.body.tracks, limit)
      })
    }
  }

  ...
}(exports));
```

Рисунок 2.13 – Класу MusicService [22]

Контейнери станів – частина системи, що змінює стан програми відповідно до стану розмови з користувачем. Потрібно щоб програма зберігала в пам’яті такі речі, як жанр музики, про який запитує ваш користувач, кількість бажаних рекомендацій, тощо. Контейнери станів зберігають дані бесіди та відстежують стан системи. На рисунку 2.14 наведено діаграму стану, яка моделює всі стани системи рекомендацій. У кожному стані є пов'язані з ним дані, які ми отримуємо від користувача. Для цього можна опитувати користувача додатковими повідомленнями.

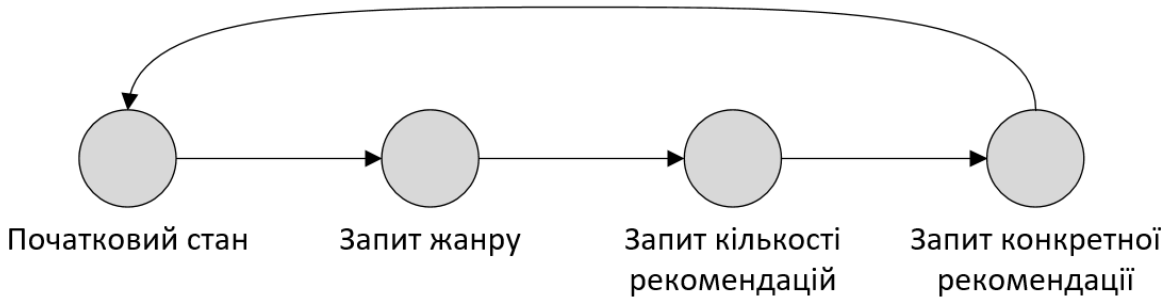


Рисунок 2.14 – Діаграма станів ЧБ для рекомендації музики

Наприклад, у стані «Запит кількості рекомендацій» повинно бути відомо жанр, який користувач вибрав. Подібним чином, стан «Запит конкретної рекомендації» повинен не тільки успадкувати дані попереднього стану, а й отримати додаткові дані про кількість рекомендацій, яку хоче користувач.

Всередині контейнеру станів міститься сутність «_users» – користувачі (рисунок 2.15). Це поле зберігає хешовану таблицю, де ключ - це ідентифікатор користувача, який ми отримуємо від повідомлення, а значення - поточний стан користувача.

```
let store = {
  _users: {},

  // creates state tree
  _initializeUserState: function(hash) {
    this._users[hash] = {
      command: '',
      spotify: {
        genre: '',
        numberOfRecs: 0
      }
    }
  },

  ...
}
```

Рисунок 2.15 – Контейнер станів [22]

Функція «initializeUserState», яка викликається першою взаємодією користувача з системою. Тут визначаються дані про стан та початковий стан. Коли потрібно оновити стан викликається функція «update» (рисунок 2.16). Вона бере старий стан та зливає його з новим станом.

До контейнеру станів також відносяться функції «getState» і «clearState» (рисунок 2.17). «getState» використовується для отримання поточного стану користувача, а «clearState» для скидання стану розмови до початкового.

```
let store = {
  ...

  update: function(hash, data) {
    if (!this._users[hash]) this._initializeUserState(hash)

    console.log('⚫  => PREVIOUS STATE')
    console.log(this.getState(hash))

    this._users[hash] = _.merge(this._users[hash], data)

    console.log('🟡  => NEXT STATE')
    console.log(this.getState(hash))
  }
}
```

Рисунок 2.16 – Функція «update» у контейнері станів [22]

```
let store = {
  ...

  getState: function(hash) {
    if (!this._users[hash]) return this._initializeUser(hash)

    return this._users[hash]
  },

  clearState: function(hash) {
    this._users[hash] = this._initializeUser()
  }
}
```

Рисунок 2.16 – Функції «getState» і «clearState» у контейнері станів [22]

Результатом є бот Aria зображений на рисунку 2.17.

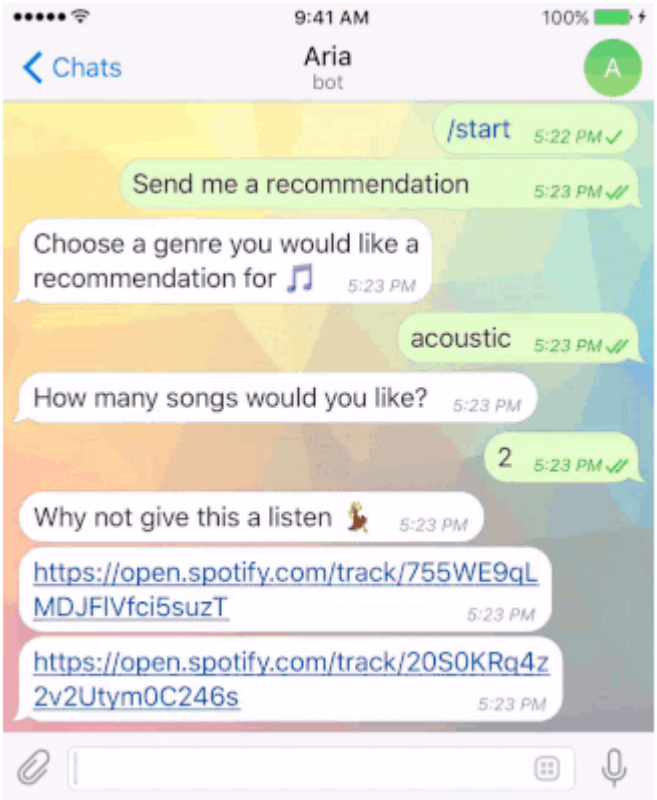


Рисунок 2.17 – Інтерфейс бота Aria [22]

Розглянута архітектура повторюється у інших ЧБ з відкритим кодом, з модифікаціями пов’язаними із використовуваною мовою програмування та стилем написання, тому вона є оптимальною [23]. Слід зазначити, що оскільки контейнер станів є просто об'єктом, він призначений для зберігання лише тимчасового стану. Якщо необхідно зберігати дані постійно, наприклад інформацію про користувача або історію покупок, то необхідно використовувати замість нього базу даних.

Описана загальна архітектура бота є частиною основних принципів їх розробки та грає важливу роль у можливості подальшого масштабування та вдосконалення. Як зазначалося, у сервісах можна використовувати засоби нейронних мереж та штучного інтелекту для покращення досвіду користувача.

Також при розширенні функціоналу необхідно враховувати можливу кількість користувачів та обирати коректні засоби та інструменти для розробки та розміщення бота в мережі. Детальніше алгоритм вибору необхідних інструментів розглядається у розділі 4.

3. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, ЇХ СКЛАДОВІ ТА ЗАСТОСУВАННЯ

Штучні нейронні мережі були побудовані за принципом біологічних нейронних мереж, які представляють собою сукупність нервових клітин, які виконують певні фізіологічні функції. Складовим елементом нейронних мереж є нейрони. На рисунку 3.1 наведено структуру людського нейрону.

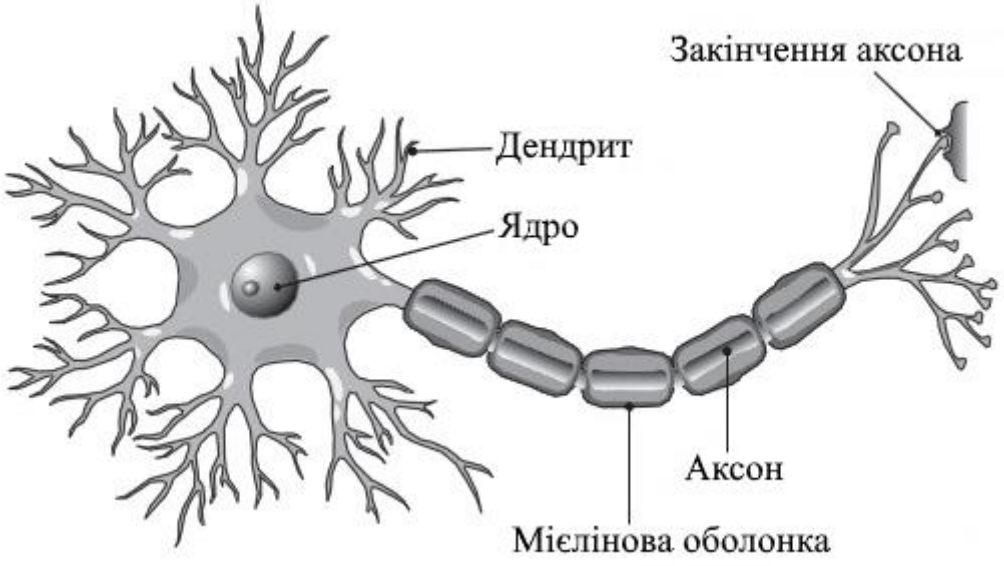


Рисунок 3.1 – Типова структура нейрона [24]

Нейрон має декілька ключових функцій:

а) Функція отримання інформації – синапси слугують джерелами інформації;

б) Інтегративна функція – на виході нейрону сигнал несе у собі інформацію про всі підсумовані в нейроні вхідні сигнали;

в) Провідникова функція – інформація передається по аксону;

г) Передавальна функція – переданий імпульс змушує медіатор на закінченні аксона передавати збудження наступному нейрону.

Синапсами називають фізичні зв'язки, якими вихідні сигнали одних нейронів надходять на входи інших. У штучних нейронних мережах кожен зв'язок характеризується певною вагою. Зв'язки, вага яких позитивна, називаються збуджуючими, а ті, вага яких негативним - гальмуючими. Вихід нейрона називається аксоном. У штучної нейронної мережі штучний нейрон – це деяка нелінійна функція,

аргументом якої, є лінійна комбінація всіх вхідних сигналів. Така функція називається активаційною. Результат активаційної функції посилається на вихід нейрона. Сукупність таких нейронів називають штучною нейронною мережею.

Функція активації нейрона характеризує залежність сигналу на виході нейрона від суми сигналів на його входах. Зазвичай функція є монотонно зростаючою і може знаходитися в області значень від -1 до 1 (гіперболічний тангенс) або від 0 до 1 (сигмоїда). Для деяких алгоритмів навчання необхідно, щоб активаційна функція була безперервно диференційованою на всій числовій осі. Назва штучного нейрону може утворюватися відповідно до його активаційної функції. Наприклад, назва «сигмоїдальний нейрон» вказує на те, що активаційна функція такого нейрону сигмоїдальна.

До основних типів активаційних функцій належать:

а) Порогова активаційна функція (функція Хевісайда)

$$f(x) = \begin{cases} 1 \text{ if } x \geq -w_0 x_0 \\ 0 \text{ else} \end{cases} \tag{3.1}$$

б) Сигмоїдальна активаційна функція

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

в) Гіперболічний тангенс

$$tanh(Ax) = \frac{e^{Ax} - e^{-Ax}}{e^{Ax} + e^{-Ax}} \tag{3.3}$$

Функцію Хевісайда не можна використовувати для алгоритму зворотного поширення помилки.

3.1 Перцептрон

Перцептрон - тип штучного нейрона, який було розроблено Френком Розенблатом в 1950-их – 1960-их роках. Хоча у сучасних роботах частіше використовують сигмоїдальну модель штучного нейрона, щоб зрозуміти, як працює сигмоїдальний нейрон, необхідно розглянути структуру і принцип роботи перцептрону. Перцептрон приймає на вхід певні значення x1, x2 ... xn, а на виході видає бінарний результат (рисунок 3.2). Розенблат запропонував використання, так званої, ваги (w1, w2, … wn) - чисел, що відображають важливість вкладу кожного із входів нейрона для кінцевого результату.

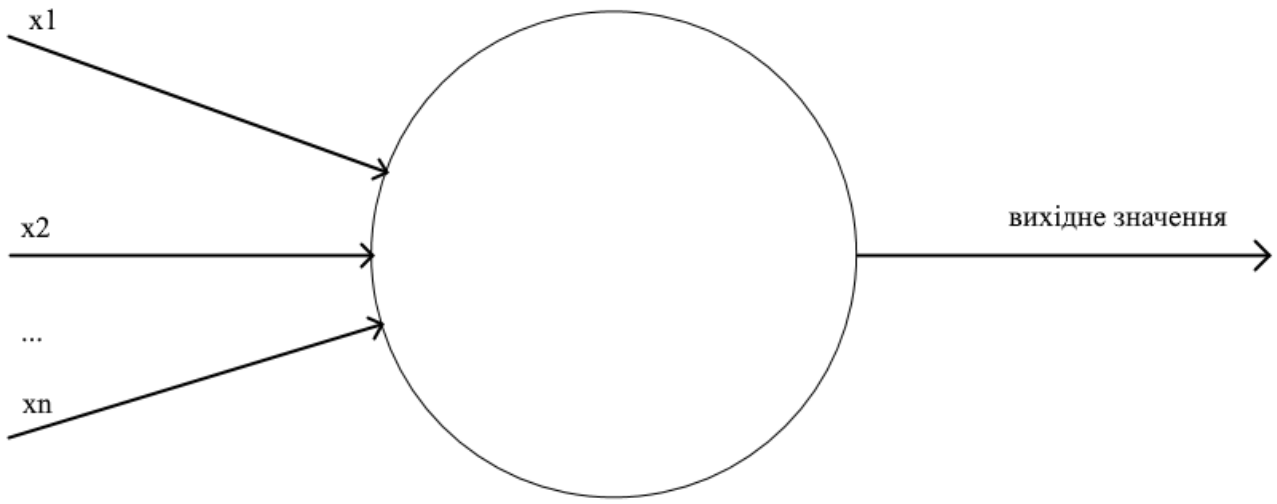


Рисунок 3.2 – Схема перцептрона [25]

Зважена сума входів порівнюється з граничним значенням і відповідно до цього на виході буде видано 0 або 1. Порогове значення є також важливим параметром нейрона.

$$\begin{cases} 0 \text{ if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 \text{ if } \sum_j w_j x_j > \text{ threshold} \end{cases} \tag{3.4}$$

Типові штучні нейронні мережі на базі таких перцептронів можна охарактеризувати такими властивостями:

а) Вони містять один прихований шар;

б) У них порогова активаційна функція;

в) У них відбувається пряме розповсюдження сигналу.

Навчання перцептрону полягає в зміні його вагової матриці.

Існують 3 історично сформованих види перцептронів [25]:

а) Одношаровий перцептрон, вхідні елементи якого безпосередньо пов'язані з вихідними за допомогою системи ваг. Є найпростішою мережею прямого поширення (feedforward network);

б) Багатошаровий перцептрон Розенблатта у якому присутні додаткові приховані шари;

в) Багатошаровий перцептрон Румельхарта у якому присутні додаткові приховані шари, а навчання проводиться за методом зворотного поширення помилки (backpropagation algorithm).

Якби певна невелика зміна ваги (або зміщення) викликала б невелику зміну на виході мережі, то бажаної поведінки нейронної мережі можна було б досягнути за допомогою простих модифікацій в процесі навчання. Проте навчання, якщо нейронна мережа складається з перцептронів, не так просто здійснити. Невелика зміна ваги або зміщення одного з перцептронів мережі може кардинально змінити його вихідне значення з 0 на 1. Тому незначні зміни значень одного з елементів мережі можуть призвести до труднощів у розумінні змін поведінки мережі. Оскільки завдання навчання нейронної мережі є завданням пошуку мінімуму функції помилки в певному просторі навчальних станів, то для його вирішення можуть застосовуватися стандартні методи теорії оптимізації. Для одношарового перцептрону з n входами і m виходами, пошук мінімуму функції необхідно буде здійснювати в nm-вимірному просторі [26].

3.2 Сигмоїдальний нейрон

Сигмоїдальні нейрони схожі на перцептрони, однак невеликі зміни в їх вагах і зсувах значно менше змінюють вихід нейрону. Цей факт дозволяє мережі, яка складається з сигмоїдальних нейронів ефективніше навчатися. На вхід сигмоїдального нейрону подаються будь-які значення від 0 до 1, так само як і у перцептрона. На виході ж видається не просто значення 0 чи 1, а певне число у проміжку між 0 і 1, оскільки активаційною функцією такого нейрону виступає сигмоїда, яка є нелінійною [27].

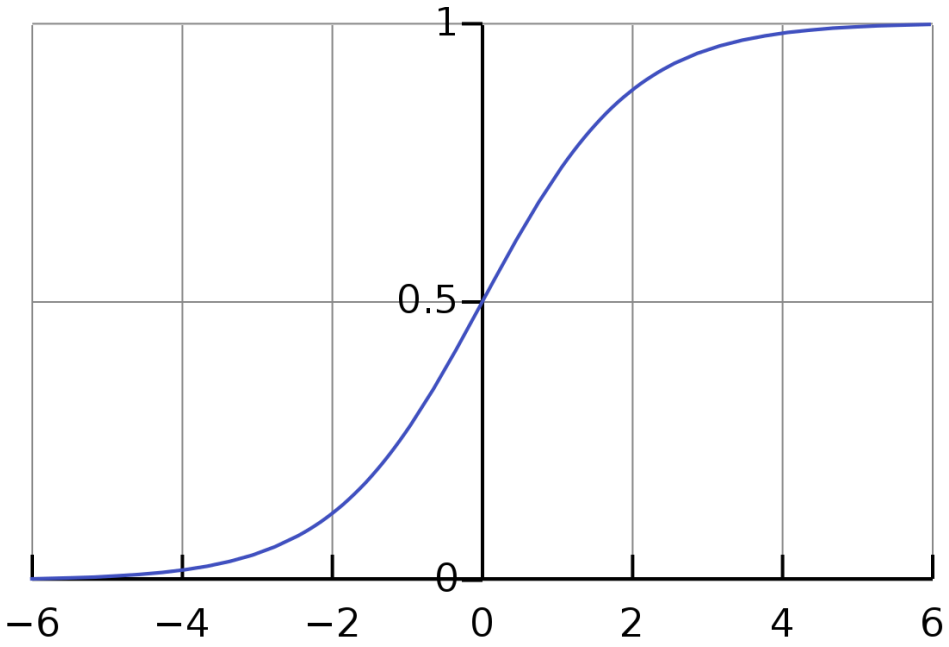


Рисунок 3.3 – Графік сигмоїди [27]

$$f(x) = \frac{1}{1 + e^{-\beta x}} \tag{3.5}$$

Чим більше β (параметр нахилу сигмоїдальної функції активації), тим більша крутизна графіка. При $\beta \to \infty$ сигмоїда наближатиметься до функції Хевісайда.

Важливою властивістю сигмоїдальної функції є її диференційованість. Застосування неперервної функції активації дозволяє використовувати при навчанні градієнтні методи.

На рисунку 3.4 наведено приклад простого нейрону реалізованого мовою C#.

```
public class SimpleNeuron {

  private double[] weights;

  public SimpleNeuron(double[] weights){
    this.weights = weights;
  }

  public double classify(double[] input){
    double value = 0.0;

    // Adding the bias
    value += weights[0] * 1.0;

    // Adding the rest of the weights
    for(int i = 0; i < input.length; i++)
      value += weights[i + 1] * input[i];

    // Passing the value through the sigmoid activation function
    value = 1.0 / (1.0 + Math.exp(-1.0 * value));

    return value;
  }
}
```

Рисунок 3.4 – Приклад нейрону на мові C# [27]

Власне функція «classify» відповідає за класифікацію даних по категоріях. Для цього прикладу було вручну вибрано та запрограмовано значення ваг, щоб забезпечити хорошу класифікацію. У наступних підрозділах буде розглянуто можливості автоматичного визначення цих значень, використовуючи деякі навчальні дані. Даний приклад ілюструє можливість створення примітивних штучних нейронів засобами сучасних мов програмування. Удосконалені модифікації таких нейронів формують шари, які в свою чергу складають ШНМ. Уже зараз можна помітити примітивну функцію класифікації, яку виконує нейрон. Отже прослідковується зв'язок з необхідністю класифікувати токени із повідомлення користувача при застосуванні ШНМ у ботах.

Нейрони можна розділити на такі групи, в залежності від їх положення в мережі:

а) вхідні нейрони, які приймають початковий вектор даних;

б) проміжні нейрони, в яких відбуваються основні обчислювальні операції при навчанні;

в) вихідні нейрони, які відображають результат роботи мережі.

3.3 Архітектура нейронних мереж

Розглянемо завдання навчання мережі. Дано безліч тренувальних прикладів X з мітками, які позначають очікуване вихідне значення мережі. Нейронні мережі визначають нелінійну гіпотезу $h_{w,b}$ (x) з параметрами w (weight – вага) і b (bias - зміщення) [28]. Нейронна мережа складена з безлічі простих нейронів так, що вихід одного з нейронів буде входом іншого (рисунок 3.5).

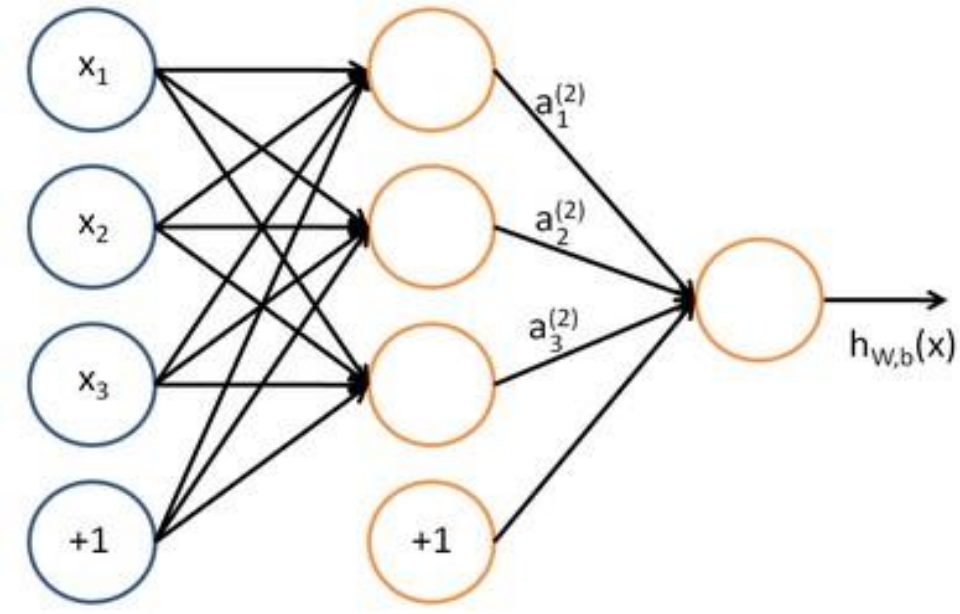


Рисунок 3.5 – Схема простої мережі прямого поширення [28]

Крайній лівий шар називається вхідним, а крайній правий шар - вихідним. Шар посередині є прихованим і називається так через те, що його значення не

відстежуються на навчальних прикладах. Таким чином в даній мережі три елементи входу, три прихованих елемента і один вихідний елемент.

Нехай n_l - кількість шарів в мережі (в даному випадку 3). Параметри мережі (w, b) = ($w^{(1)}$, $b^{(1)}$, $w^{(2)}$, $b^{(2)}$). Результат застосування функції активації (виходу) позначається a_i для кожного i-ого елемента [29]. Отримаємо систему:

$$\begin{aligned}&\{a_1^{(2)} = f(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b^1_{(1)}),\\ &a_2^{(2)} = f(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b^1_{(2)})\}\end{aligned} \tag{3.6}$$

Позначивши функцію суми через z, отримаємо у векторній формі:

$$\begin{cases} z^{(2)} = W^{(1)}x + b^{(1)} \\ a^{(2)} = f(z^{(2)}) \\ z^{(3)} = W^{(2)}a^{(2)} + b^{(2)} \\ h = f(z^{(3)}) \end{cases} \tag{3.7}$$

Тоді загальна формула системи матиме вигляд:

$$\begin{cases} z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \\ a^{(l+1)} = f(z^{(l+1)}) \end{cases} \tag{3.8}$$

Мережею прямого поширення називаються нейронні мережі, які використовують вихід одного шару в якості вхідних даних для наступного шару.

3.4 Навчання нейромереж

3.4.1 Загальні поняття в навчанні нейронних мереж

Спершу необхідно дати визначення загальним поняттям перед дослідженням принципів навчання нейронних мереж.

Епоха - прямий і зворотний прохід по усіх тренувальних прикладах.

Розмір партії (batch) - кількість тренувальних прикладів для однієї ітерації прямого і зворотного проходів.

Кількість ітерацій - кількість проходів, при цьому кожен прохід використовує партії (batch). Одним проходом вважається сума прямих проходів та зворотних проходів по цій партії. Наприклад, маючи 1000 тренувальних прикладів, розмір партії рівний 500, буде потрібно дві ітерації, щоб завершити одну епоху.

З математичної точки зору, навчання нейронних мереж - багатопараметричне завдання нелінійної оптимізації.

3.4.2 Алгоритм зворотнього поширення помилок

Алгоритм зворотнього поширення помилки визначає стратегію підбору ваг багатошарової мережі із застосуванням градієнтних методів оптимізації. Оскільки цільова функція, яка зазвичай визначається як сума квадратичних різниць між фактичними і очікуваними вихідними значеннями, є безперервною, градієнтні методи оптимізації є високоефективними при навчанні мережі. За такого методу навчання багатошарової нейронної мережі необхідно обчислити вектор градієнта щодо параметрів всіх шарів мережі.

Нехай є кінцевий набір тренувальних даних (m прикладів). Для навчання нейронної мережі застосуємо пакетний градієнтний спуск (batch gradient descent). Квадратична помилка цільової функції (squared-error cost function) для одного прикладу буде обчислена за формулою 3.9

$$J(W,b;x,y) = \frac{1}{2}||h_{W,b}(x) - y||^2 \tag{3.9}$$

Тоді цільова функція, для m прикладів, буде виглядати так:

$$\begin{aligned} J(W,b) &= [\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 \\ &= [\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2)] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 \end{aligned} \tag{3.10}$$

Перший член виразу – J(W, b) – це сума квадратів помилок, другий - член регуляризація (L2 - зменшення ваг - weight decay), що дозволяє зменшити значення ваг і запобігти перенавчанню. Параметр регуляризації ваг λ використовують для перевірки відносної значущості частин даного виразу. У завданнях бінарної класифікації *y* представлений нулем або одиницею (оскільки сігмоїдна функція видає значення в межах від 0 до 1, проте при використанні функції активації на базі гіперболічного тангенса проміжок був би від -1 до 1). Основне завдання при навчанні – це мінімізація J(W, b). Для навчання нейронної мережі необхідно ініціалізувати кожен параметр *i* випадковими величинами, близькими до нуля, а потім застосувати алгоритм оптимізації (згаданий вище градієнтний спуск) [29].

Оскільки J(W, b) не є опуклою функцією, то градієнтний спуск сприйнятливий до локальних оптимумів. Кожна ітерація градієнтного спуску оновлює параметри таким чином:

$$\begin{aligned} W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b) \\ b_i^{(l)} &= b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b), \end{aligned} \tag{3.11}$$

Тут α – швидкість навчання.

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)}) \right] + \lambda W_{ij}^{(l)} \quad (3.12)$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x^{(i)},y^{(i)}) \quad (3.13)$$

Далі опишемо кроки алгоритму зворотного поширення помилки:

а) Здійснюється прямий прохід по мережі, обчислюються активації шарів L2, L3 і так далі, аж до вихідного шару L_{nl};

б) Для кожного вихідного елементу *i* в вихідному шарі n_l (the output layer) розраховується помилка

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \quad (3.14)$$

в) Для $l = n_l - 1, n_l - 2, n_l - 3, \ldots$; Для кожного елемента в шарі *l*, розраховується

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \quad (3.15)$$

г) Вираховуються часткові похідні:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)} \quad (3.16)$$

$$\frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)} \quad (3.17)$$

3.4.3 Деталі використання градієнтного спуску

Основними недоліками градієнтного спуску при навчанні мережі є параліч мережі та визначення оптимального розміру кроку.

Параліч мережі наступає в результаті встановлення значень ваг мережі, в результаті корекції, у дуже великі значення. Оскільки помилка, що посилається назад в процесі навчання, пропорційна похідній стискаючої функції, то процес навчання може практично зупинитися. Цьому можна запобігти, зменшуючи крок η, однак процес навчання відбуватиметься довше.

Також проблемою є визначення оптимального кроку для спуску. Якщо значення кроку не змінюється і воно досить мале, то метод сходитиметься занадто повільно. Якщо ж крок занадто великий, то може виникнути параліч мережі. Необхідно змінювати значення кроку: збільшувати до тих пір, поки не припиниться поліпшення оцінки в напрямок антиградієнта і зменшувати, якщо оцінка не поліпшується.

Якщо для пакетного градієнтного спуску функція втрат обчислюється для всіх зразків разом узятих після закінчення епохи, а потім змінюються вагові коефіцієнти нейронів, то для стохастичного методу вагові коефіцієнти змінюються після обчислення виходу мережі на кожному з навчальних прикладів. Недоліком пакетного градієнтного спуску є можливе його «застрягання» в локальних мінімумах.

Незважаючи на те, що стохастичний метод працює повільніше пакетного, він здатний виходити з локальних мінімумів, що призводить до кращих результатів навчання мережі (стохастичний метод використовує недовирахуваний градієнт).

Для покращення ефективності навчання використовують функцію перехресної ентропії в якості функції втрат, де $y`_i$ – це передбачені значення, а y_i – вірні значення [29].

$$L(x,y) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log a(x^{(i)}) + (1-y^{(i)}) \log\Big(1 - a(x^{(i)})\Big) \qquad (3.18)$$

Також для покращення ефективності навчання нейронної мережі можливо застосовувати такі методи регуляризації:

a) L1-регуляризація;

b) L2-регуляризація;

c) Викидання (dropout);

d) Штучне розширення даних для навчання.

При використанні L1-регуляризації відбувається зміна нерегуляризованої цільової функції шляхом додавання суми абсолютних значень ваг:

$$J = J_0 + \frac{\lambda}{n} \sum_W |W| \tag{3.19}$$

За такого методу відбувається прямування одного або декількох значень ваг до 0.0, тому відповідна функція більше не буде потрібною. Цей ефект називається селекцією функцій (feature selection).

Використання L2-регуляризації (також відомої як weight decay) на відміну від L1-регуляризації, припускає зменшення ваг на величину, пропорційну до них:

$$J = J_0 + \frac{\lambda}{2n} \sum_W W^2 \tag{3.20}$$

Метод регуляції Dropout не впливає на значення цільової функції. За його використання змінюється сама структура мережі. Кожен нейрон видаляється з мережі з певною ймовірністю *p*. У отриманій прорідженій мережі відбувається зворотнє поширення помилки, для решти ваг робиться градієнтний крок. Після цього вилучені нейрони відновлюють в мережі. При навчанні нейромережі вихід кожного нейрона домножується на (1-p). Так буде отримано математичне очікування відповіді мережі по її 2N (де N - кількість нейронів в мережі) архітектурах. Навчена таким чином мережа є результатом усереднення 2N мереж. Наприклад, окрема нейронна мережа,

навчена за допомогою ранньої зупинки, має дуже велику помилку, однак усереднення декількох таких нейронних мереж призводить до істотного зниження помилки.

3.4.4 Основні параметри при навчанні мережі

Наведемо основні параметри на які варто звертати увагу при навчанні мережі. Одним із основних є темп навчання. Для практичного його застосування спочатку необхідно оцінити граничне значення для η, в якому значення цільової функції миттєво починає знижуватися без коливань. Спочатку значення оцінки встановлюється $\eta = 0.01$. Якщо значення цільової функцій знижується під час перших епох, то потрібно збільшувати темп навчання, поки не буде знайдено значення коливання цільової функції. Якщо ж при початковому темпі навчання значення цільової функції коливаються, то необхідно його зменшувати. Темп навчання регулює розмір кроку в градієнтному спуску в залежності від значення цільової функції, визначаючи, чи не був розмір кроку градієнтного спуску занадто великим.

Використання ранньої зупинку (early stopping) застосовується для визначення розміру епох навчання. Рання зупинка означає, що в кінці кожної епохи потрібно обчислити точність класифікації на перевірочних даних (validation set). Коли поліпшення точності припиниться необхідно зупинити процес навчання. Така зупинка також запобігає перенавчанню [29].

Графік навчання полягає у ідеї – зберігати темп навчання незмінним до моменту, поки точність даних перевірки не почне погіршуватися. Тоді необхідно зменшити темп навчання (наприклад, в 10 разів).

Розмір пакетів – теж надзвичайно важливий параметр при навчанні мережі. Якщо розмір пакетів занадто малий, неможливо повністю використовувати переваги хороших матричних бібліотек, оптимізованих для швидкого обладнання. Якщо ж розмір пакетів занадто великий, то ваги мережі будуть оновлюватися не достатньо часто. Необхідно вибрати компромісне значення, яке максимізує швидкість навчання.

3.5 Глибокі нейронні мережі

Глибокими нейронними мережами називаються такі мережі, в яких є кілька прихованих шарів. Оскільки кожен прихований шар обчислює нелінійне перетворення попереднього шару, глибока мережу може мати значно більшу репрезентативну потужність (тобто може представляти значно складніші функції), ніж мережі із малою кількістю шарів. При навчанні глибокої мережі важливо використовувати нелінійну функцію активації в кожному прихованому шарі. Це пов'язано з тим, що безліч шарів лінійних функцій поодинці вираховували б тільки лінійну функцію введення, а, отже, не були б точнішими, ніж за використання тільки одного прихованого шару [30].

Головним достоїнством глибоких мереж є стисле представлення достатньо великої множини функцій. Можна показати, що існують функції, які k-шарова мережа може представляти стисло, а (k-1)-шарова - не змогла б цього зробити, хіба якщо вона мала б експоненціально велику кількість елементів в прихованих шарах.

За допомогою методу, описаного вище, можна покладатися тільки на марковані дані для навчання. Однак помічених даних часто буває недостатньо, а, отже, для багатьох завдань важко отримати достатню кількість прикладів для відповідності параметрам складної моделі. Наприклад, з огляду на високий ступінь виразності глибоких нейромереж, навчання при невеликій кількості даних призведе до перенавчання.

Навчання мережі із малою кількістю шарів (наприклад, з 1 прихованим шаром) з використанням контрольованого навчання зазвичай призводить до зближення параметрів з відповідними значеннями. Але при навчанні глибокої мережі, це працює набагато рідше. Зокрема, навчання нейронної мережі з використанням навчання з прикладами включає в себе вирішення проблеми з неопуклою оптимізацією (наприклад, мінімізація помилки навчання, в залежності від параметра мережі W).

$$\sum_i ||h_W(x^{(i)}) - y^{(i)}||^2 \qquad (3.21)$$

У глибокій мережі з'являється велика кількість локальних оптимумів, тому навчання з градієнтним спуском перестає працювати. При використанні методу зворотного поширення помилки для обчислення похідних, градієнти, які поширюються від вихідного шару до більш ранніх шарів мережі, швидко зменшуються в міру збільшення глибини мережі. В результаті похідна від загальної вартості по відношенню до ваги в більш ранніх шарах дуже мала. Таким чином, при використанні градієнтного спуску ваги ранніх шарів змінюються повільно, а ранні шари навчитися значно повільніше. Цю проблему часто називають «дифузією градієнтів» (diffusion of gradients) [30].

3.5.1 Проблеми навчання глибоких мереж і їх рішення

Проблема зникаючого градієнта – це складність, яка виникає при навчанні штучних нейронних мереж з використанням методів навчання на основі градієнта і зворотного поширення помилки. В таких методах кожна вага в нейронній мережі оновлюється пропорційно градієнту функції помилки щодо поточної ваги на кожній ітерації навчання. Стандартні функції активації, такі, як гіперболічний тангенс, мають градієнти в діапазоні (-1, 1), а метод зворотного поширення помилки обчислює їх за ланцюговим правилом. Далі відбувається множення цих чисел для обчислення градієнтів «фронтальних» шарів в n-шарової мережі, що означає, що градієнт (сигнал помилки) експоненціально зменшується разом з n, а передні шари навчаються дуже повільно. Коли використовуються функції активації, похідні яких можуть набувати великих значення, є ризик зіткнутися з проблемою вибухаючого градієнта (exploding gradient problem). Далі наведемо можливі вирішення цієї проблеми.

Багаторівнева ієрархія: шар мережі попередньо навчається, використовуючи методи навчання без нагляду, а потім його значення регулюється за допомогою методу зворотного поширення помилки. Таким чином кожен шар мережі має стисле уявлення про те, що подається на наступний шар [30].

Довга короткострокова пам'ять – один із різновидів архітектури рекурентних нейронних мереж. Коли величини помилки розповсюджуються в зворотному

напрямку від вихідного шару, помилка не випускається з пам'яті LSTM-блоку. Вона безперервно передається назад кожному з нейронів, поки ті не будуть навчені відкидати подібні значення[].

Залишкові мережі (Residual networks, ResNets) – це один з найбільш ефективних методів вирішення проблеми зникаючого градієнта. Глибша мережу матиме вищу помилку навчання, ніж мережа з малою кількістю шарів. Команда Microsoft Research виявила, що поділ глибокої мережі на частини (скажімо, кожна частина являє собою тришарову мережу) та передача вхідних даних в кожен фрагмент, перед передачею їх наступному, допомогли, практично повністю, усунути проблему зі зникненням градієнту. Ніяких додаткових параметрів або змін в алгоритмі навчання не потрібно. ResNets показали нижчу помилку навчання (і тестову помилку), ніж їх аналоги із меншою кількістю шарів, уникаючи проблеми зникаючого градієнта, за допомогою поділу на дрібніші підмережі.

Використання сигмоїдальних активаційних функцій може викликати проблеми в навчанні глибинних мереж, а саме значення активацій в кінцевому шарі будуть близькі до нуля на ранніх етапах навчання, сповільнюючи цей процес. Були запропоновані альтернативні активаційні функції, які не так програють у часі від такого обмеження.

Вибір відповідних ваг і стохастичний градієнтний спуск на основі імпульсу (momentum-based stochastic gradient descent) значно впливають на здатність глибокої мережі до навчання[30].

4. РОЗРОБКА

Для розробки програми необхідно спершу визначити певні засоби та інструменти які використовуватимуться. До таких можна віднести:

а) Мова програмування;

б) Середовище розробки;

в) Засоби хостингу застосунків;

г) Система контролю версій;

д) CI/CD системи;

е) Додаткові засоби та інструменти.

У цьому розділі буде проаналізовано можливості вибору таких засобів та наведено приклад розробки бота за їх допомогою.

4.1 Процес створення бота та огляд Telegram Bot API

Для створення бота у системі Телеграм існує бот – BotFather, на рисунку 4.1. Необхідно розпочати з ним діалог та виконати декілька простих кроків. Основною ціллю цієї процедури є отримання токена авторизації бота, який дозволяє серверам Телеграм унікально ідентифікувати бота. Цей токен буде використовуватися програмою при будь яких запитах до сервера [31].

Після початку роботи із ботом BotFather потрібно використати команду «/newbot», для створення нового бота. BotFather попросить вас вказати ім'я бота та ім'я користувача, перед тим як створити токен авторизації.

Ім'я бота – це те, що відображатиметься у контактній інформації користувачів та в багатьох інших місцях. Ім'я користувача для бота – це коротке ім'я, яке буде використовуватися у згадуваннях та посиланнях сайту telegram.me. Імена користувачів повинні містити від 5 до 32 символів. Вони нечутливі до регістру, але можуть включати тільки латинські символи, цифри та підкреслення [31]. Ім'я бота повинно закінчуватися на «bot», наприклад, «PersonalizedInfoBot» або «personalized_info_bot».

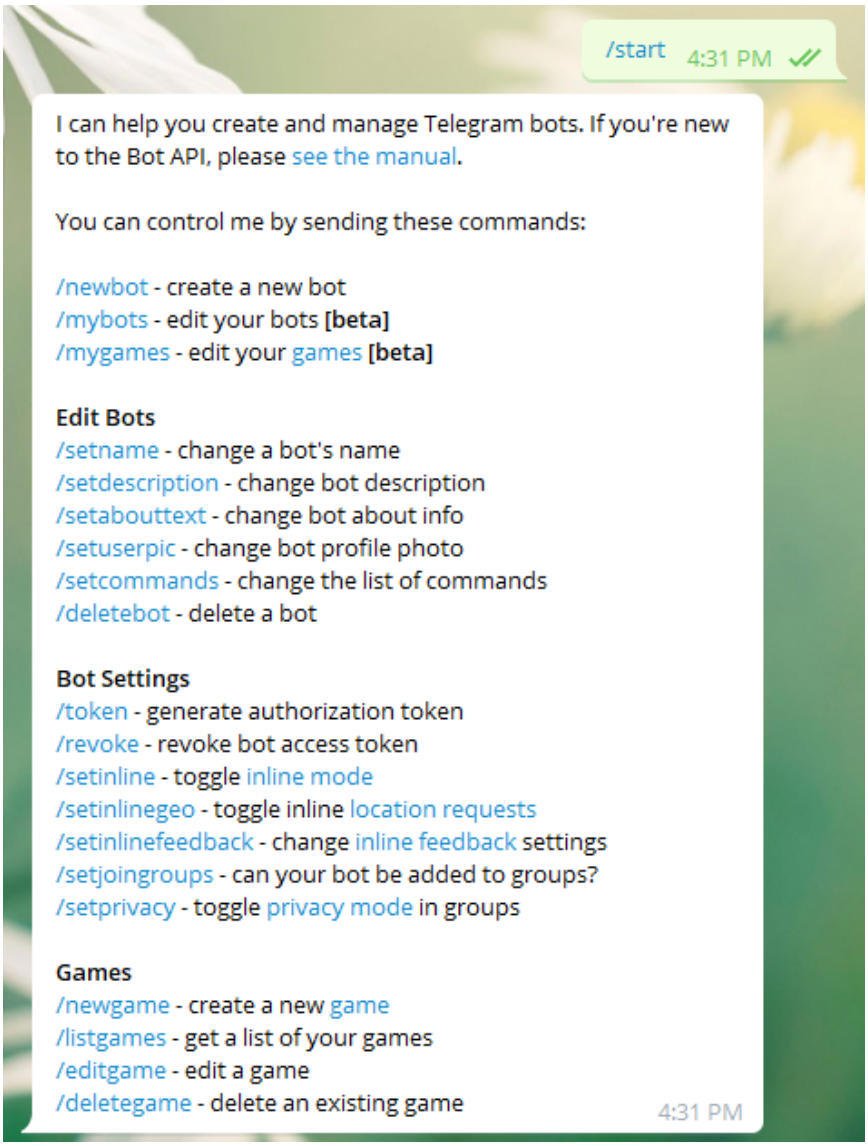


Рисунок 4.1 – Початок роботи із ботом BotFather та доступні команди [32]

Не слід недооцінювати важливість імені бота, оскільки воно завжди видиме користувачу і фактично асоціюється із компанією. Зазвичай ботів називають так як і компанію (лише з приставкою «bot»), або комбінують назву компанії із функцією яку виконуватиме бот, наприклад: «Domino's Pizza Bot» чи «Amazon Search & Price Bot». Діалог обрання імені бота та імені користувача для бота, наведено на рисунку 4.2.

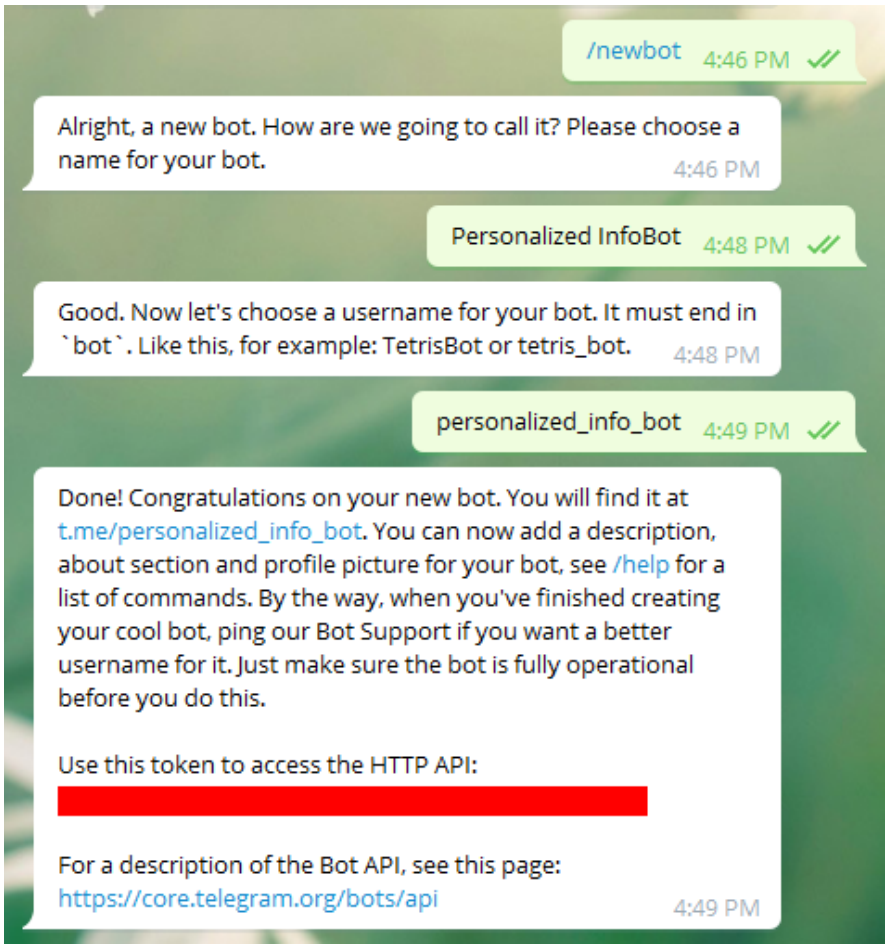


Рисунок 4.2 – Процес отримання токену за допомогою BotFather

BotFather – це також інструмент для керування уже створеними ботами. Список доступних команд для редагування ботів складається із:

а) /mybots – повертає список ботів та дозволяє редагувати їх налаштування за допомогою зручних елементів керування;

б) /mygames – повертає список ігор та дозволяє редагувати їх налаштування за допомогою зручних елементів керування;

в) /setname – дозволяє змінити ім’я бота;

г) /setdescription – дозволяє змінити опис бота – короткий текст до 512 символів, що описує його суть. Користувач бачить цей текст перед початком розмови з ботом під заголовком «Що може зробити цей бот?»;

д) /setabouttext – дозволяє змінити інформацію про бота – короткий текст до 120 символів. Користувачі бачать цей текст на сторінці профілю бота

(рисунок 4.3). Коли вони діляться своїм ботом з кимось, цей текст надсилається разом із посиланням;

е) /setuserpic – дозволяє змінити зображення профілю бота. Зображення бота відображається в пошуку, діалозі та списку контактів користувача, а тому важливо серйозно поставитися до його вибору;

ж) /setcommands – дозволяє змінити список команд, що підтримуються ботом. Користувачі побачать ці команди як пропозиції, коли вони вводять «/» в чаті. Кожна команда має назву, яка повинна починатися з косої риски «/», містити букви, цифри чи підкреслення та бути коротше 32 символів, параметри та текстовий опис;

з) /deletebot – дозволяє видалити бота. При цьому його ім’я користувача стає доступним для використання іншими;

и) /setinline – дозволяє ввімкнути чи вимкнути режим використання у будь-якому чаті для бота;

к) /setinlinegeo – дозволяє боту використовувати дані про місцезнаходження користувача, щоб забезпечити результати на основі місцезнаходження;

л) /setjoingroups – дозволяє змінити можливість додавання бота до груп. Будь-який бот повинен мати можливість обробляти приватні повідомлення, але якщо він не призначений для роботи в групах необхідно відключити цю опцію;

м) /setprivacy – дозволяє визначити які повідомлення бот отримуватиме при роботі в групах. Якщо «/setprivacy» вимкнено – бот отримуватиме всі повідомлення.

Після отримання токену можна приступати до подальшого процесу розробки, та слід пам’ятати ключові технічні відмінності ботів від користувачів:

а) У ботів немає статусу який відображає чи бот в мережі та відповідно відсутня позначка часу коли бот востаннє був в мережі. Натомість усі боти мають підпис «bot» під іменем, як, наприклад, у вікні опису бота «markerBot» на рисунку 4.3;

б) У ботів обмежена місткість хмарного сховища повідомлень, а тому застарілі повідомлення будуть видалятися із сервера, як тільки вони були прочитані користувачем. Проте програми можуть зберігати цю інформацію на власних серверах;

в) Боти не можуть самі розпочати діалог із користувачем. Користувач повинен або добавити їх у групу, або першим написати особисте повідомлення. Користувачі можуть використовувати посилання telegram.me/<ім’я бота> або пошук для того щоб розпочати діалог із ботом;

г) Імена користувачів-ботів закінчуються на «bot»;

д) Після того як їх додали до групи, боти, за замовчуванням, не отримують усі повідомлення із неї. Це можливо змінити у налаштуваннях.

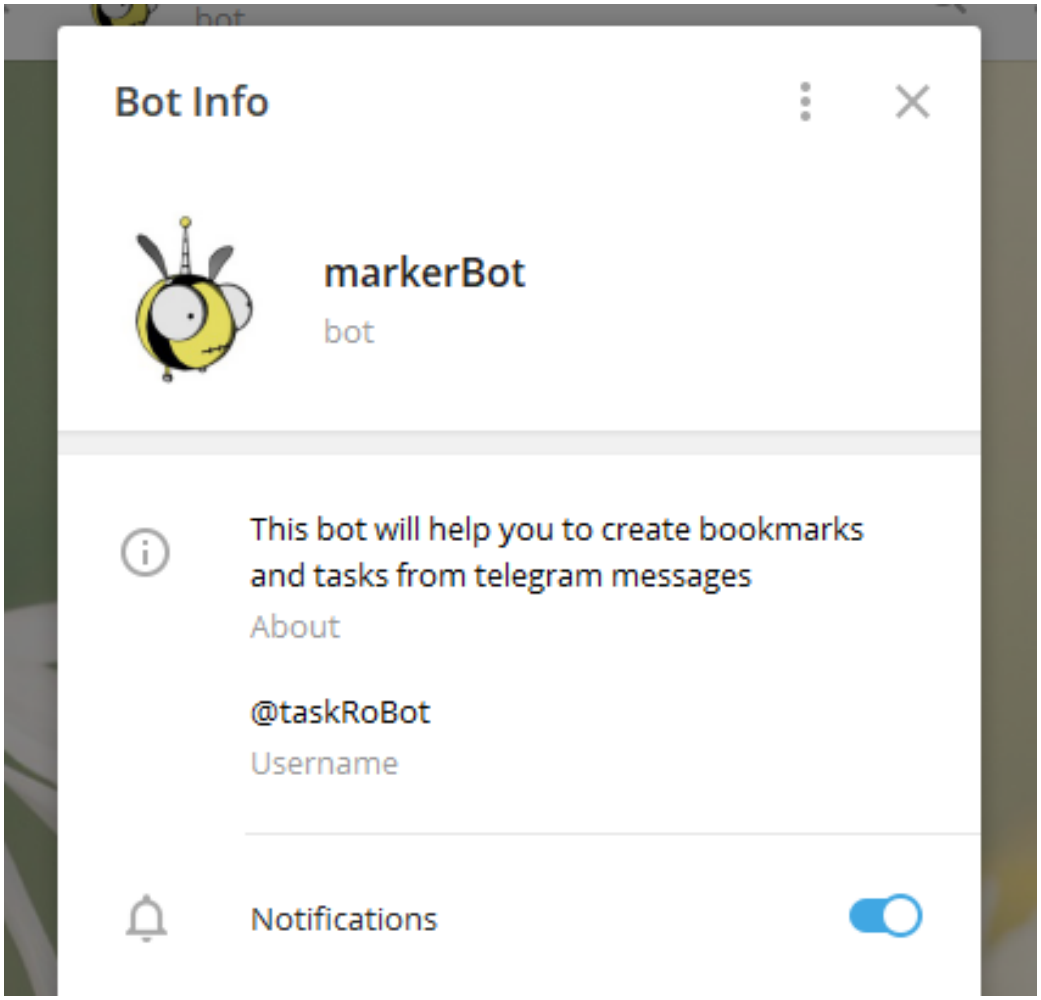


Рисунок 4.3 – Вікно інформації про бота

Перед тим як обирати інструменти розробки, необхідно ознайомитися із прикладним програмним інтерфейсом (англ. Application Programming Interface, API), який надає месенджер для розробників. Прикладний програмний інтерфейс - набір визначень взаємодії різнотипного програмного забезпечення[33]. Якщо простіше – це рівень абстракції необхідний для зручної комунікації декількох програм, при якій одна або обидві з них надають доступ до певних високорівневих функцій іншим, не розкриваючи внутрішньої їх реалізації. В той же час програма яка використовує API отримує доступ до функціоналу програми, яка надає його, без необхідності детально вивчати структуру та зв’язок її елементів. Інтерфейси прикладного програмування полегшують для розробників використання певних технологій при створенні додатків. Також API дозволяють приховати внутрішні особливості реалізації методів для покращення безпеки та забезпечення можливостей для зовнішньої взаємодії із програмою.

У попередніх розділах ми прийшли до висновку що найкращим варіантом для створення чат-бота є месенджер Телеграм. Телеграм пропонує два види API для розробників. Перше - це API Telegram, який призначений для створення сторонніх клієнтів для платформи Телеграм. В даній роботі він використовуватися не буде. Другий - Bot API, що дозволяє легко створювати програми, які використовують повідомлення Телеграм як інтерфейс у спілкуванні із користувачем.

Власне Bot API – це те, що дозволяє легко та зручно створювати чат-ботів. Цей API дозволяє підключати ботів до системи Телеграм. З точки зору системи Телеграм, боти – це спеціальні облікові записи, для яких не потрібно встановлювати додатковий номер телефону. Ці облікові записи служать інтерфейсом для коду, що працюватиме на нашому сервері [32].

Для цього не потрібно нічого знати про те, як працює протокол шифрування MTProto, який застосовується у системі Телеграм – їх проміжний сервер буде обробляти всі шифрування та спілкування з API Телеграм. Для зв'язку з цим сервером використовується простий HTTPS-інтерфейс, який пропонує спрощену версію API Телеграм.

Будь-які запити до Bot API Телеграм повинні передаватися через протокол HTTPS і мати наступну форму: https://api.telegram.org/bot<ТОКЕН>/<ІМЯ МЕТОДУ>. На місці <ТОКЕН> відповідно повинен міститися токен авторизації бота, а на місці <ІМЯ МЕТОДУ> – метод який необхідно використати з API.

В API підтримуються GET і POST HTTP запити. GET-запит використовується за необхідності отримання певної інформації від API, а GET-запит – при необхідності передати якусь інформацію засобами Bot API [32].

Відповідь на будь-який запит містить об'єкт JSON, який завжди має логічне поле «ok», яке вказує на успішність запиту, а також може мати необов'язкове поле «description» з описом результату. Якщо «ok» встановлено в «true», запит був успішним, а результат запиту можна знайти в полі «result». У випадку невдалого запиту «ok» містить «false», а помилка описується в полі «description». Поле числового типу «error_code» вказує на код помилки. Деякі помилки також можуть мати необов'язкове поле «parameters», яке може допомогти автоматично опрацювати помилку.

Є два взаємовиключні способи отримання оновлень для бота – метод «getUpdates» і використання вебхуків, обробників подій. У випадку використання вебхуків сервер Телеграм, за наявності неопрацьованих оновлень, спровокує подію з якою пов'язаний веб-хук, а той відреагує на неї та отримає необхідні оновлення. Вхідні оновлення зберігаються на сервері до тих пір, поки бот не отримає їх у будь-який із вище описаних способів, але не більше 24 годин. Отже, необхідно щонайменше раз на 24 години отримувати вхідну інформацію. Насправді, це необхідно робити набагато частіше, оскільки частиною цієї інформації є повідомлення користувачів.

Розглянемо основні типи які підтримує Bot API, та які можуть бути передані як у запиті бота, так і при відповіді на цей запит:

а) User – цей об'єкт являє собою користувача або бота;

б) Chat – цей об'єкт являє собою чат чи групу;

в) Message – цей об'єкт являє собою повідомлення. Використовується як для надсилання інформації користувачу, так і для отримання від нього відповідей;
г) MessageEntity – один із елементів повідомлення (наприклад тег, ім’я користувача чи url-посилання);
д) PhotoSize – відображає розмір фотографії, ескізу файлу чи стікера;
е) Audio – аудіо файл, який можна програвати за допомогою месенджера;
ж) Document – файл будь-якого не описаного типу;
з) Video – описує відео файл;
и) Voice – голосове повідомлення надіслане користувачем;
к) VideoNote – відео повідомлення надіслане користувачем;
л) Contact – описує контакт телефонної книги;
м) Location – відповідає певній точці на карті;
н) Venue – описує певне місце проведення події;
о) UserProfilePhotos – зображення профілю користувача;
п) File – файл, який можна завантажити. Гарантовано, що посилання буде дійсним протягом щонайменше години. Коли термін дії посилання закінчиться, можна згенерувати нове за допомогою команди «getFile». Максимальний розмір файлу для завантаження - 20 Мб;
р) А також – команди пов’язані зі створенням індивідуальної клавіатури, відповідями від такої клавіатури, надсиланням різноманітних медіа-файлів, а також пов’язані з роботою із елементами та користувачами чатів.

Відповідно до наведених типів існують методи пов’язані із їх надсиланням та отриманням. Також існують методи для більш зручної роботи із чатами – вони дозволяють, наприклад, змінювати елементи, такі як фото, опис, та ін., заблокувати користувача чату чи розблокувати його та багато інших корисних функцій.

Важливо що Телеграм, як платформа, дозволяє ботам проводити різноманітні операції як із повідомленнями, так і з чатами, а також підтримує та навіть заохочує використання ботів для різноманітної взаємодії. Факт того, що для покращення

взаємодії із користувачами, вони обрали бота, як інтерфейс для створення інших ботів, тільки підтверджує те, що боти стали невід'ємною частиною сучасного спілкування. Крім того, як частина месенджера доступний бот для надсилання стікерів – спеціальних картинок, які заміняють emoji.

Гнучкість, зумовлена наявністю великої кількості доступних функцій, дозволяє персоналізувати ботів під потреби різних замовників. Для використання бота можна використовувати команди, повідомлення, інтерактивні клавіатури чи навіть голосові повідомлення. Відповіді користувачу можна наводити у формі текстових повідомлень, фото, відео чи файлів.

4.2 Вибір мови програмування та середовища розробки

Мову програмування можна обирати довільно, проте спершу необхідно дослідити чи існують доступні реалізації API обраною мовою програмування і, якщо ні – або обрати іншу мову, або розробити свою реалізацію API, що, звісно, збільшить затрати на розробку.

Для написання усіх компонентів системи обрано сучасну, високотехнологічну мову об'єктно-орієнтованого програмування від компанії Microsoft під назвою C#. Вона була розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research. Обрана мова програмування відноситься до сім'ї мов з C подібним синтаксисом, а саме таких як C++ або Java. Програмне забезпечення, розроблене на мові C# налаштоване на платформу Microsoft .NET Framework. Мова має строгу типізацію даних, підтримує поліморфізм, наслідування, тощо.

Мова програмування C# є високорівневою мовою і за часи свого становлення та розвитку ввібрала, та продовжує вбирати в себе позитивні риси із інших мов. Так C# увібрав позитивні риси своїх попередників таких, як C++, Delphi, Pascal та ін. Також дана мова швидко розвивається та регулярно оновлюється, вносячи все новий і новий функціонал, не рідко запозичений із інших популярних нині мов програмування, до прикладу: Java, Python, Perl, та ін [34].

До вагомих переваг C# належить наявність якісно розроблених вбудованих бібліотек, що розповсюджуються у складі .NET Framework чи .NET Core. Зараз Microsoft працює над створенням нової структури бібліотек, яка забезпечуватиме більшу гнучкість для розробників.

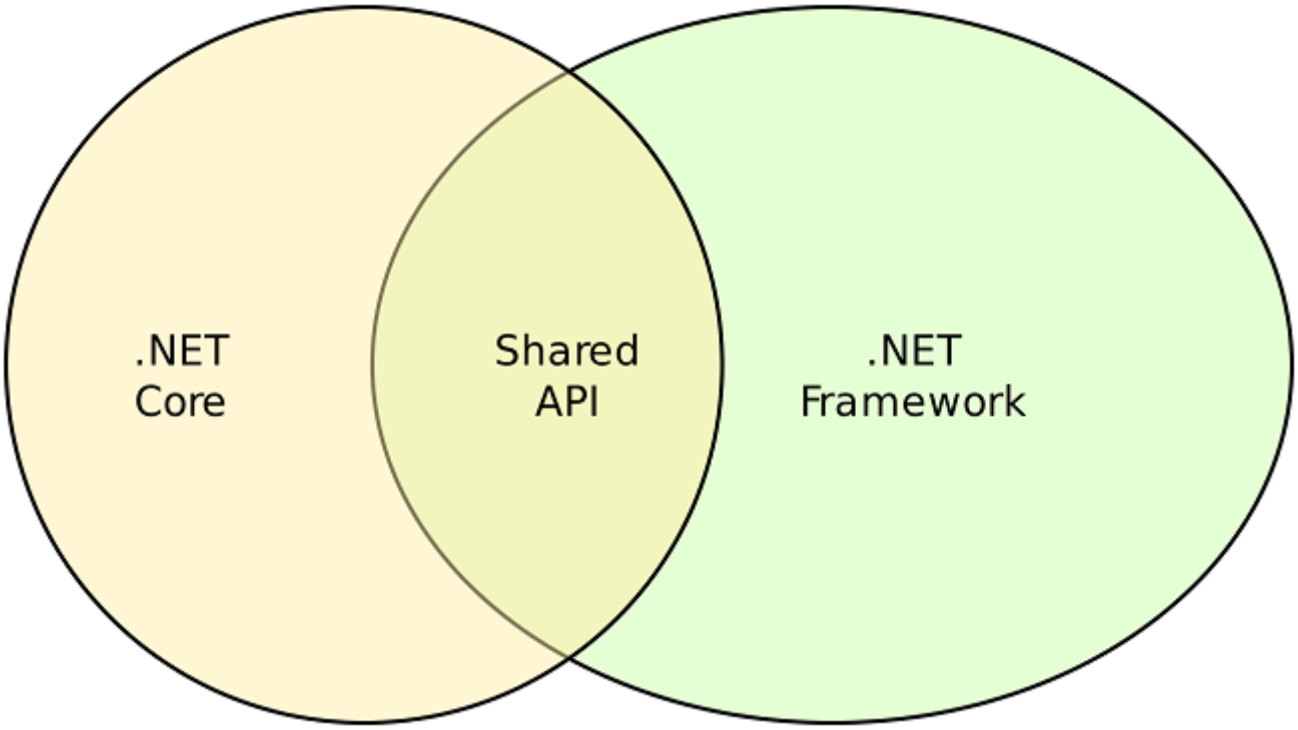


Рисунок 4.4 – Взаємозв’язок між .NET Framework та .NET Core [35]

Доки не відбувся такий поділ використовуватимемо .NET Framework, як перевірену часом збірку бібліотек. Серед основних бібліотек в ній можна виділити [34]:

а) всі базові перетворення між строковим, числовим і логічним типами, що реалізуються класом Convert (System);
б) колекції (System.Collections);
в) класи пов’язані із роботою з Xml (System.Xml);
г) мережеві класи (System.Net);
д) багатопоточність (System.Threading);
е) класи для роботи з процесами та діагностикою продуктивності при налагодженні програм (System.Diagnostics);

Також, є деякі корисні вбудовані речі в мові C#, яких немає в багатьох інших мовах:

а) вбудована конструкція foreach, яка дозволяє перераховування елементів колекцій;

б) автоматичний збирач сміття, за використання якого немає потреби вручну звільняти пам'ять, а вивільнення пам’яті відбувається дуже швидко, також не потрібно вручну керувати пам'яттю;

в) властивості – замість того, щоб писати методи доступу, можна писати властивості, які об'єднують в собі два методи доступу і не мають параметрів. Все стає більш лаконічно та особливо позначається на красі інтерфейсу класу;

г) все, що успадковане від класу object може бути перетворено в рядок методом ToString(), який можна перевизначити на свій розсуд.

C# виявляє і обробляє помилки допущені розробником на етапі компіляції, перериваючи її, якщо ці помилки впливають на роботу програми або виводячи попередження, якщо не впливають. Дана функція підвищує ефективність розробника та дозволяє покращувати якість написаного програмного продукту.

Серед помилок які виявляє компілятор мови C#:

а) повернення значення не у всіх умовних гілках, або відсутність повернення значення;

б) використання невизначеної змінної;

в) неявне перетворення більшого за розміром в ОЗУ числа в менше за розміром, тобто double в float або long в int чи short чи byte і т.п.;

г) неявне перетворення знакового і беззнакового числа одне в одного: int в uint, short в ushort і т.п .;

д) неявне перетворення bool в число, чи навпаки;

е) перестрибування з однієї мітки case на іншу, якщо перша не закрита;

ж) командою переривання break і при цьому щось містить або є останньою;

з) звернення до елементу із недопустимим рівнем приватності;

и) та ін.

Серед попереджень можна зазначити:

а) не використана змінна, функція, подія (приватна або локальна);

б) недосяжність коду – коли перед ним стоїть команда переходу (return, goto, break, continue).

Слід зазначити, що наведені вище перерахування включають в себе лише малу частину можливостей такого потужного інструменту як C#.

Можливості мови програмування C# тісно пов’язані із виконуючим середовищем байт-кодів CLR (Common Language Runtime). Особливості мови диктуються тим, яка версія CLR встановлена і що вона може транслітерувати, а що ні. Враховуючи тісний зв’язок цих компонентів, з розвитком виконуючого середовища – розвивається й мова програмування, так як можливості зростають і надається все більше сервісів, які полегшують розробку.

За історію розвитку C# відбувалося значні й не значні зміни від однієї версії до іншої. Таким чином, для підтримки найбільш сучасних, а також для забезпечення й ще підтримуваних операційних систем було обрано версію 6.0, що надає підтримку для платформу .NET Framework 6.0.

Програмування на мові C# можна здійснювати у багатьох IDE(Integrated Development Environment — Інтегроване середовище розробки):

а) Visual Studio 2015, 2017;

б) Visual Studio Code (спрощене середовище розробки доступне на різних операційних системах);

в) MonoDevelop (середовище розробки доступне на різних операційних системах);

г) Sharpdevelop (середовище з відкритим кодом);

д) Notepad++;

е) та ін.

Найпоширенішим та найзручнішим середовищем розробки програмного забезпечення на мові C# визнається Microsoft Visual Studio. Воно доступне у декількох версіях:

а) Community – безкоштовна повнофункціональна версія IDE;
б) Professional – надає професійні інструменти для розробників, а також послуги та підписки для невеликих команд;
в) Enterprise – комплексне рішення для великих команд розробників.

Будь-який розробник може створювати безкоштовні або платні програми за допомогою Visual Studio Community. Visual Studio Community 2017 містить всі функції Visual Studio Professional 2017, що призначені і оптимізовані для індивідуальних розробників, учнів, учасників проектів з відкритим кодом і невеликих груп [36].

Visual Studio Community можна використовувати згідно наступних правил:

а) може використовувати необмежена кількість користувачів в організації в наступних випадках: в навчальних аудиторіях, для наукових досліджень або участі в проектах з відкритим кодом;
б) в некорпоративних організаціях Visual Studio Community може використовувати до 5 користувачів;
в) в корпоративних організаціях (в яких використовується більше 250 ПК або річний дохід яких перевищує 1 млн доларів США) використання заборонено, за винятком випадків, перелічених вище (відкритий код, наукові дослідження та навчальні аудиторії).

Обрано Community версію продукту, оскільки, на даному етапі, вона задовольняє усі потреби. У майбутньому, при збільшенні кількості людей в команді варто розглянути версію Professional.

Для роботи із Telegram Bot API обрано .Net клієнт із відкритим кодом «Telegram.Bot»[37]. Ця реалізація має хорошу документацію, є актуальною, постійно поповнюється новим функціоналом та доступна для установки із вбудованого у Visual Studio менеджера пакетів «NuGet Package Manager».

Також для розробки бота необхідні засоби ASP.NET – інструменту від Microsoft для розробки веб-додатків. ASP.NET пропонує три фреймворки для створення веб-додатків: Web Forms, ASP.NET MVC та ASP.NET Web Pages. Веб-додатки можливо

створювати за допомогою будь-якого із них, проте у всіх є свої особливості. Web Forms та ASP.NET Web Pages орієнтовані на створення веб-сторінок та веб-сайтів, в той час, як ASP.NET MVC дозволяє реалізувати веб-додаток без HTML інтерфейсу. Саме цей фреймворк і обрано для подальшої розробки.

4.3 Вибір хостинг провайдера

Далі обирається хостинг провайдер, на серверах якого власне розгортатиметься бот. Також можна обрати варіант із розміщенням застосунку на власних серверах. Слід враховувати як ціну, так і можливості масштабування ресурсів хостингу, за потреби, наприклад, якщо одночасно потрібно буде обробляти запити великої кількості користувачів. Варіант розміщення застосунку на власних серверах доцільно розглядати або за наявності уже робочого сервера на якому є доступні ресурси, або за необхідності підвищення безпеки, хоча сучасні провайдери хостингу пропонують спеціальні тарифи, за таких потреб. У будь-якому випадку, хостинг на віртуальних виділених серверах (virtual private server – VPS) надає необхідну гнучкість та достатню безпеку.

Для обрання провайдера проаналізуємо наявні на ринку пропозиції. Серед ключових компаній які надають такі послуги – Microsoft із платформою Azure, Amazon із Amazon Web Services та Heroku. Усі вони надають необхідну функціональність та різноманітні API для роботи із серверами. Важливим фактором при виборі хостинг провайдера є можливість інтеграції із іншими продуктами, що використовуються при виконанні проекту, наприклад, IDE та засобами неперервної інтеграції, розгортання та доставки.

Проаналізувавши таблицю 4.1, як хостинг провайдера обрано Microsoft Azure. Оскільки цей продукт дешевший за інші та має низку переваг, зокрема інтеграцію із Visual Studio 2017, та можливість використання інших продуктів Microsoft, наприклад Microsoft Cognitive Services.

Таблиця 4.1 – Порівняння хостинг провайдерів

	Хостинг провайдер		
Критерій	Microsoft Azure	Amazon Web Services	Heroku
Наявність API для роботи із сервером	Так	Так	Так
Інтеграція з IDE	Так	Ні	Ні
Інтеграція з CI/CD	Так	Так	Так
Базова ціна за місяць	18$ (~470 грн)	37$ (~970 грн)	50$ (~1300 грн)

4.4 Обрання засобів неперервної інтеграції, розгортання та доставки

Спершу необхідно ознайомитися із цими термінами. Неперервна інтеграція – це практика розробки програмного забезпечення, яка полягає в злитті віток розробників в загальну основну гілку розробки кілька разів на день, а також виконанні частих автоматизованих збірок проекту для якнайшвидшого виявлення потенційних дефектів та рішення інтеграційних проблем[38].

Оскільки стадія інтеграції є останньою вона займала невизначений обсяг часу, через те, що на цьому етапі виникали помилки, які було важко чи навіть неможливо виявити раніше. Через це розробка продукту могла значно затриматися. На зміну такому підходу прийшла неперервна інтеграція, за якої етап інтеграції відбувається щоразу як вносяться зміни у основну вітку розробки або якщо необхідно, за певних інших умов. За умови роботи команди, можна легко відслідкувати які саме зміни призвели до виникнення помилок. Це дозволяє виявити можливі помилки одразу після розробки певної частини програмного забезпечення, а тому здешевити їх подальше усунення.

Неперервна доставка – це засіб, який гарантує можливість розгортання(доставки) нового функціоналу користувачам у будь-який момент і за мінімально можливий час. Проте, не варто вважати, що це означає доставку функціоналу користувачам, як тільки він був розроблений та протестований.

Неперервне доставка – це гарантія того, що цей функціонал працюватиме у користувача саме так, як це задумувалося розробниками та менеджерами.

Неперервне розгортання – це наступний крок після неперервної доставки. Кожна зміна, що проходить усі попередні етапи та проходить автоматичні тести, автоматично розгортається у користувача. Постійне розгортання повинно бути метою більшості компаній, які не обмежуються регуляторними чи іншими вимогами.

Варто зазначити, що існують правові випадки, за яких необхідно чекати, поки функція можна перейти до розгортання у користувача, що робить постійне розгортання непрактичним. Наприклад, за умови розробки програмного забезпечення для медичного обладнання, необхідно отримати дозвіл на постачання розроблених функцій від регулюючої організації. У кожному окремому випадку необхідно визначатися чи потрібна ця функціональність при розробці, чи вона становитиме лише додаткові витрати.

Усі ці інструменти базуються на трьох основних принципах, які доводять їх ефективність. Перший принцип – розмежування відповідальності зацікавлених сторін:

а) Розробники та дизайнери відповідають за зовнішній вигляд, логіку та досвід використання продукту, відповідно до вимог користувачів, і несуть відповідальність за створення робочих функцій, підтвердженням чого є модульні тести;

б) Інженери з якості (QE чи QA) відповідають за належну якість продукції, тестуючи функцій по мірі їх розробки і проводять випробування, щоб упевнитися, що клієнт отримає продукт, який працює без помилок;

в) Бізнес-аналітики (BA) і власники продуктів (PO) несуть відповідальність за прийнятність розробок, наприклад, їх вартість для бізнесу. Це підтверджується взаємодією з фактичними користувачами і створенням історій користувачів – завдань які описують їх бажання щодо того чи іншого функціоналу. Вони координують і аналізують результати тестів, та перевіряють відповідність уже розробленого функціоналу вимогам

користувача. Також до їхньої відповідальності відноситься пріоритизація завдань, відповідно до їх значущості для клієнтів;

г) Операційний відділ (DevOps) – несуть відповідальність за доступність продукту користувачам. Працюючи в області CI / CD, вони масштабують концепції в міру необхідності і розробляють логістику коду, щоб код від розробників міг перейти в виробниче середовище, а користувачі могли легко отримувати доступ до нових функцій;

д) Користувачі несуть відповідальність за використання продукту. Мається на увазі, що продукт повинен бути корисний і виправдовувати зусилля.

Кожен етап CI/CD створює середовище, налаштоване на те, щоб групи, які відповідальні за певний етап у життєвому циклі програмного забезпечення, могли легко отримати доступ до всіх інструментів, що необхідні для виконання покладених на них функцій. Відповідно забезпечується цілісність процесу розробки та доставки продукту.

Другий принцип – зниження ризику. Кожен етап було задумано для зниження ризику в певному аспекті. Розробники відповідають за логіку роботи програми і написаного коду, щоб знизити ризик порушення логіки. QE відповідають за цілісність досвіду користувачів, а також за написання тестів направлених на зниження ризику помилок при реалізації історій користувачів. BA і PO відповідають за зручність використання і беруть участь в тестах на відповідність функціоналу вимогам користувачів, щоб знизити ризик створення непридатних та небажаних функцій. DevOps беруть участь в обслуговуванні всього процесу розробки та відповідають за розгортання і масштабування систем, щоб знизити ризик тривалої недоступності продукту.

Оскільки етапи призначені для зниження конкретного ризику, важливо відзначити, що кожен етап рівноцінно важливий для визначення кінцевої готовності певного функціоналу.

Третій принцип – мінімізація часу зворотного зв'язку. Причина впровадження CI/CD – використання машин для роботи з людьми. Це дозволяє скоротити час, що витрачається на зворотний зв'язок щодо функцій, які ще тільки розробляються. За

допомогою масштабування скорочується час, що витрачається на тестування програмного забезпечення, що дозволяє автоматизувати процес розгортання. Ці процедури займуть набагато більше часу, якщо будуть виконуватися вручну.

Однак не варто відкидати важливість людей при розробці програмного забезпечення. Наприклад, ніколи на даному етапі розвитку технологій автоматизувати тести, пов’язані зі зручністю використання програм, неможливо.

Важливо також пам’ятати, що неефективна автоматизація завжди буде потребувати більшого вкладання ресурсів та капіталу, ніж взагалі її відсутність.

До основних інструментів пов’язаних із CI/CD можна віднести:

а) Система відслідковування помилок та управління проектами;

б) Систему контролю версій;

в) Сервер для автоматизованої збірки проекту (білд-сервер).

Оберемо систему відслідковування помилок та управління проектами. Є велика кількість таких систем, які призначені для різної кількості користувачів та різної направленості проектів. Серед основних можна виділити:

а) Basecamp;

б) OpenProject;

в) Trello;

г) Atlassian JIRA;

д) Redmine.

Basecamp – найпростіша система, призначена для управління широким спектром проектів (рисунок 4.5). Проте у проектах пов’язаних із розробкою програмного забезпечення її використовують рідко, оскільки вона має дуже обмежений функціонал. Зокрема в ній не існує поняття «баг» чи помилка, а описати завдання можна лише як пункт у списку завдань. Також дана система є платною, а вартість тарифу необхідного для роботи декількох користувачів сягає 999 доларів США на рік. Проте очевидною її перевагою є простота використання та вбудований чат.

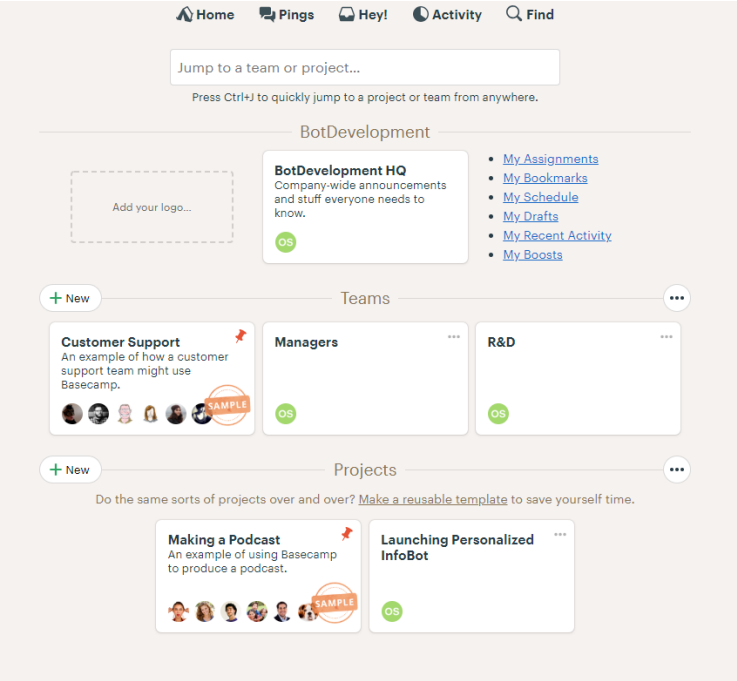


Рисунок 4.5 – Головна сторінка компанії у Basecamp

OpenProject – веб-додаток який дозволяє керувати проектами (рисунок 4.6). До основних його функцій можна віднести:

а) Управління проектами та вітками;

б) Відстежування помилок;

в) Сторінка опису проекту;

г) Можливість розміщення документів;

д) Наявність форума, для спілкування між користувачами.

У даного продукту існує безкоштовна версія для проектів із відкритим кодом, проте необхідна для комфортної розробки версія коштуватиме 4.95€ на місяць.

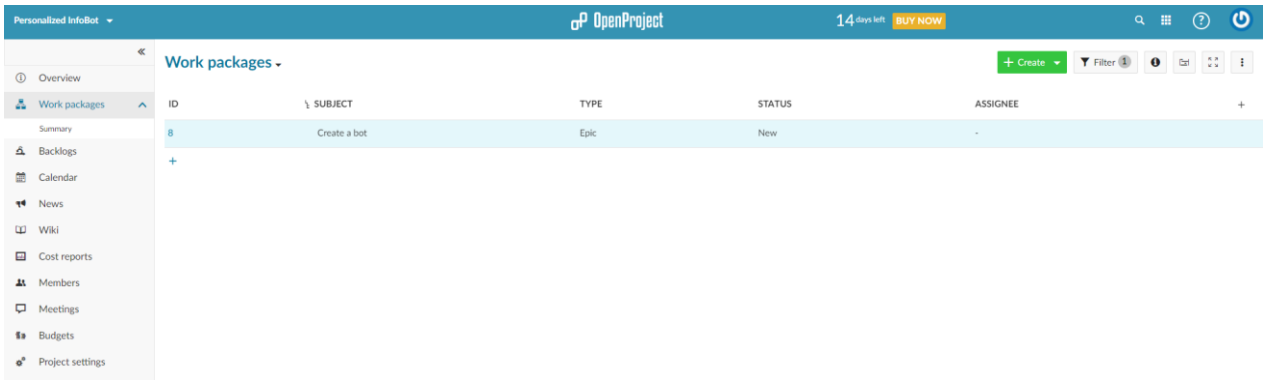


Рисунок 4.6 – головна сторінка проекту у системі OpenProject

Trello – безкоштовна система для управління проектами. Для керування проектом використовуються дошки зі списком завдань на них. При цьому завдання можна переміщати із одної дошки на іншу. Це дозволяє слідкувати за переміщенням завдання протягом усього циклу розробки. На рисунку 4.7 наведено приклад проекту із п’ятьма дошками:

а) «To do» – дошка для завдань, які необхідно зробити;

б) «In development» – дошка для завдань, які зараз розробляються;

в) «Testing» – дошка для завдань, які проходять стадію тестування;

г) «On review» – дошка для завдань, які зараз на стадії перевірки;

д) «Done» – дошка для завдань, які зроблено.

Для того щоб завдання було переміщено в колонку «Done» необхідно щоб воно пройшло усі попередні етапи.

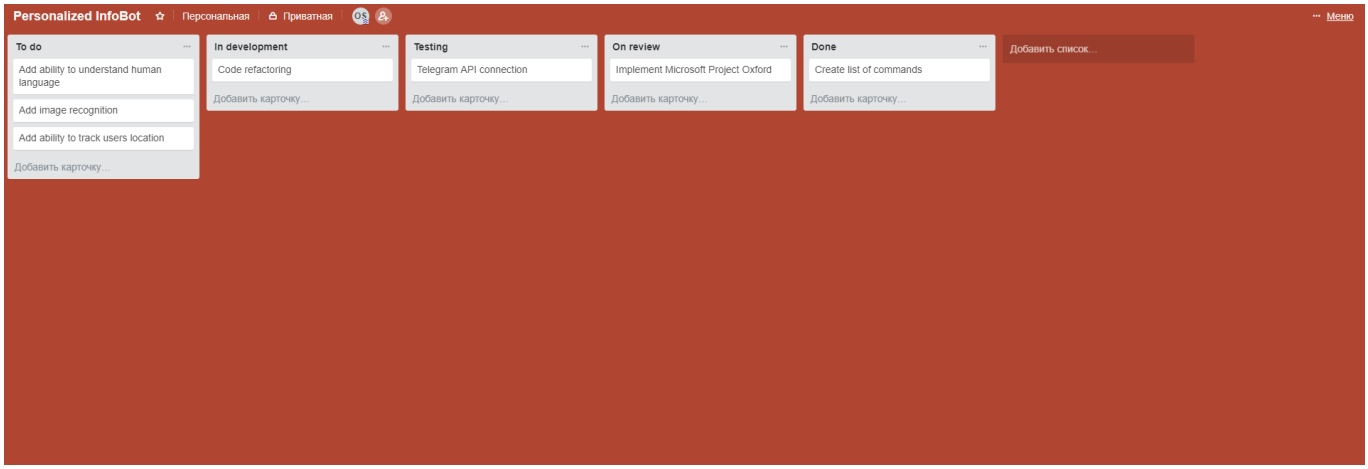


Рисунок 4.7 – Головна сторінка проекту у Trello

Для розробки даного продукту було обрано саме цю систему, проте за умов значного масштабування вона стане неефективною, а тому потрібно розглянути більш гнучкі варіанти. Одним із таких є Redmine (рисунок 4.8) – безкоштовний веб-додаток із відкритим кодом, який розробляється мовою Ruby та розповсюджується на умовах GPL (загальної публічної ліцензії), що робить цей продукт безкоштовним та дозволяє виділяти власні ресурси на його вдосконалення. Це дуже потужний продукт, та, у порівнянні із JIRA (рисунок 4.9), у нього є декілька відмінностей:

а) У ньому не можна настроювати під себе робочий процес;

б) Відсутні функції перетягування для пріоритизації;

в) Відсутність можливості формувати графіки без додаткових плагінів;

г) Наявність у JIRA магазина плагінів із більш ніж 1000 товарів;

д) Наявність у Redmine русифікованого інтерфейсу.

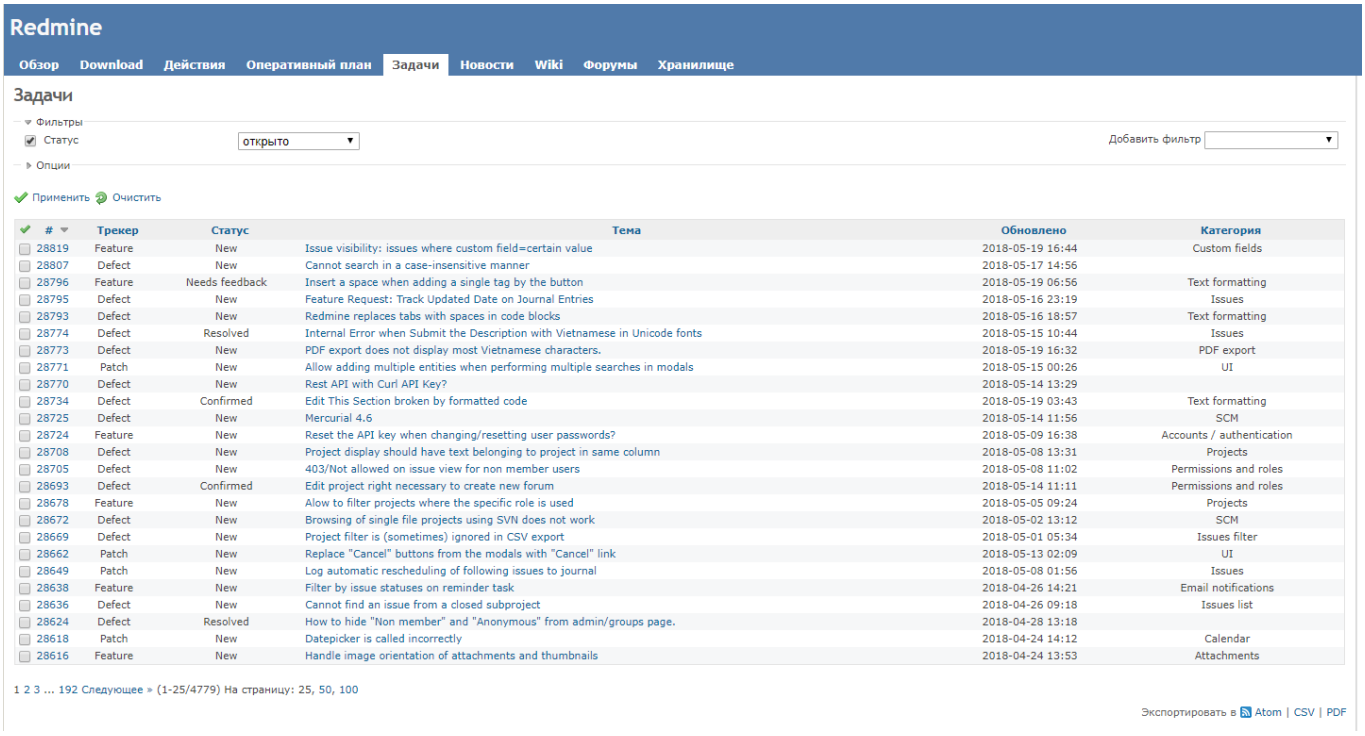


Рисунок 4.8 – Сторінка завдань у системі Redmine

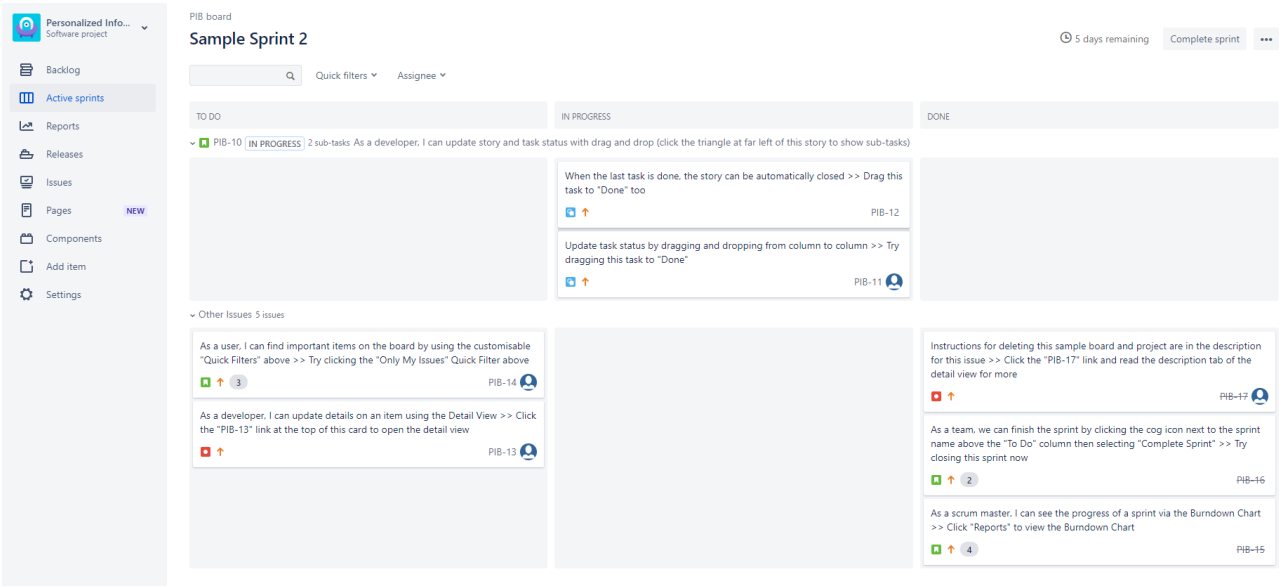


Рисунок 4.10 – Сторінка спринта проекту у системі JIRA

Далі необхідно обрати систему контрою версій. Система контролю версій – це програмний інструмент для керування версіями файлів та одиниць інформації. Серед головних існуючих рішень варто виділити GitHub, GitLab та Team Foundation Server (TFS) від Microsoft.

Таблиця 4.2 – Порівняння систем контрою версій

Функція	GitHub	GitLab	TFS
Надання різноманітних рівнів доступу	Частково обмежено	+	+
Вбудовані засоби CI/CD	–	+	–
Відслідковування завдань	+	+	+
Імпорт та експорт даних	+	+	+
Приватні репозиторії	Платно	+	–
Ціна за користувача за рік	250$	39$	36$

Колосальною перевагою для GitLab, у даному випадку, є вбудовані засоби CI/CD, що значно спрощує роботу розробника, та дозволяє на початкових етапах відмовитися від DevOps інженерів в команді. Крім того, у вартість цього продукту входить можливість створювати приватні репозиторії, які будуть доступні, як для доступу, так і для перегляду, лише користувачам яким ми надамо доступ. Для розробки обрано GitLab, як систему контролю версій та як інструмент для неперервної інтеграції та неперервної доставки. Також GitLab частково реалізуватиме функціонал сервера для автоматизованої збірки проекту. На даному етапі немає необхідності у функціях неперервного розгортання.

4.2 Застосування нейронних мереж на прикладі Microsoft Cognitive Services

Оскільки завданням даної роботи була розробка сервісу для тематичного аналізу інформації для платформи Телеграм, постала необхідність обрати інструмент, який дозволятиме, власне, аналізувати надіслані користувачем повідомлення та виділяти із них ключову інформацію. Для цих цілей необхідно застосовувати засоби штучного інтелекту та нейронних мереж, проте не завжди доцільно тратити на це ресурси. Натомість, можна скористатися уже готовими рішеннями, які дозволяють значно прискорити та полегшити розробку. Одним із таких доступних рішень є Microsoft Cognitive Services. У рамках цього продукту компанією Microsoft надається доступ до багатьох функцій штучного інтелекту, наприклад:

а) Комп’ютерний зір;
б) Розумний пошук;
в) Аналіз тексту;
г) Сприйняття голосових команд;
д) Побудова складних баз знань.

Основною метою цієї дисертації є огляд процесу розробки бота та демонстрація можливостей покращення досвіду користувача, за допомогою використання засобів нейронних мереж, тому на даному етапі зосередимося на використанні функціоналу пов’язаного із аналізом тексту. З його допомогою можна виділити ключові слова із певного уривку. В нашому випадку це використовується для аналізу повідомлення користувача, і надання йому відповіді, що включатиме інформацію про ключові слова передані у повідомленні.

Додаткові технології, які постачаються у складі Microsoft Cognitive Services можна добавити як новий функціонал для бота за подальшого розвитку проекту.

Наведемо доступний список команд без використання засобів нейромереж:

а) /help – виводить допоміжний текст;
б) /about – виводить опис бота;

в) /showinfo (параметер) – команда що відповідає за обробку параметра та виведення відповідної інформації про нього, якщо така доступна.

При використанні Microsoft Cognitive Services можна позбутися команд і надсилати боту прості повідомлення. Наприклад попередні команди можна замінити схожими на такі:

а) What can I do? – Що я можу зробити?

б) Tell me about this bot – запит на отримання опису бота.

Інші команди можна писати просто текстом. Наприклад, «I will go to Rome next weekend and I will need to visit Alfredo» – із цієї команди бот виділить ключові слова та фрази:

а) Rome (Рим)

б) Next weekend (Наступних вихідних)

в) Alfredo (Альфредо)

Відповідно до цього, він запропонує переглянути детальну інформацію про Рим, забронювати готель на наступні вихідні у Римі та зателефонувати Альфредо для попередження про прибуття.

Як бачимо це легка у використанні та доступна технологія яка надає можливість значно покращити досвід користувача при використанні бота.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Основною метою роботи є розробка бота для платформи Telegram, який надаватиме користувачу можливості зручного персоналізованого пошуку інформації. Зокрема застосунок зможе надавати інформацію про країни, міста, страви, готелі, музику, та ін. Дослідження, проведене у роботі, було сфокусоване на можливості персоналізації та адаптації бота під кожного конкретного користувача, а також на можливості коректно реагувати на його повідомлення.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
	1. Пошук користувачем корисної інформації	Зручний інтерфейс отримання інформації
	2. Аналіз повідомлення користувача надання індивідуально підібраної інформації	Відповідно до запиту та до попередніх знань про користувача буде надано індивідуально підібрану інформацію
	3. Адаптація під конкретного користувача з часом	Чим частіше користувач використовує бота, тим краще його відповіді

Конкуренти:

a) Wikipedia Bot
b) Otouto

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*			*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Wikipedia Bot*	*Otouto*			
1.	Аудиторія, вік	16 – 40	12 – 45	18 – 35	Певна обмеженість вікової групи		
3.	Перспективи росту	Високі	Низькі	Середні			Хороші перспективи зростання
4.	Функціональність	Частково обмежена	Обмежена	Частково обмежена		Хороша функціональність	
5.	Сприйняття команд	+	+	+		Як і в інших	
6.	Виділення команд з повідомлення	+	–	–			Присутня можливість
7.	Розуміння контексту	У перспективі	–	–			Перспектива удосконалення

Відповідно до Таблиці 5.2 можна виділити наступні сильні сторони проекту:

a) Високі перспективи для росту аудиторії, особливо якщо урахувати можливе розширення вікових лімітів
b) Можливість виділення команд із повідомлення

с) В перспективі – можливість розуміти контекст діалогу.

Для покращення конкурентоспроможності продукту необхідно розширити вікові рамки цільової аудиторії. Це можна зробити за допомогою надання нових функцій чи вдосконалення уже існуючих.

5.1.1 Технологічний аудит ідеї проекту

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Самописний бот	Власна розробка	Наявні	Так
2	Використання платформи для створення ботів	Покупка	Наявна	Ні
3	Модифікація уже існуючої інтеграції бота	Покупка та власна розробка	Наявна	Так
Обрана технологія реалізації ідеї проекту: Обрано реалізацію самописного бота, через оптимальне співвідношення гнучкості вихідного продукту до витрачених зусиль.				

5.1.2 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	7
2	Загальний обсяг продаж, грн/ум.од	3000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Витрати на рекламу
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	14

Хоча на ринку і присутньо багато конкурентів він є привабливим для входження, оскільки користувачі прагнуть отримати більше нового функціоналу, а затрати пов’язані із виходом на ринок не є високими.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп*	*Вимоги споживачів до товару*
Зручний інструмент тематичного пошуку	Молодь, користувачі месенджерів	Зручний інтерфейс. Підтримка задання команд повідомленнями	- Зручний інтерфейс - Красивий дизайн

Таблиця 5.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Велика к-ть конкурентів	Конкуренти ускладнюють вихід на ринок	Запропонувати недоступні у конкурентів функції.
2	Мінливі вимоги користувачів	Бажання користувачів постійно змінюються	Запропонувати зручний інтерфейс користувача для повідомлення бажаного функціоналу
3	Інфляція	Високий рівень інфляції в країні	Грамотна реалізація фінансової звітності

Таблиця 5.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Високий попит	В даній галузі існує високий попит на ботів із сучасними функціями	Необхідно запропонувати користувачу якомога сучасніші функції
2	Висока швидкість розробки	Можлива вища швидкість розробки, у порівнянні із повноцінною програмою	Необхідно знайти баланс між кількістю функцій бота та часом його розробки

5.1.3 Аналіз пропозиції: загальні риси конкуренції на ринку

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства*
1. Тип конкуренції	Монополістична	Вхідні і вихідні бар'єри існують, але невисокі. Акцентування уваги на унікальності продукту та його перевагах
2. За рівнем конкурентної боротьби	Національна	Можливість реалізації продукту по всій території України
3. За галузевою ознакою	Міжгалузева	Орієнтованість на аудиторію молодих людей
4. Конкуренція за видами товарів.	- товарно-родова – можливість користуватися простим пошуком - товарно-видова – наявність альтернативних ботів - між бажаннями – боти із різною кількістю функцій	Проведення маркетингової компанії. Приваблення користувача великою кількістю функцій

Продовження таблиці 5.8

5. За характером конкурентних переваг	Нецінова	реалізація зручного інтерфейсу взаємодії з ботом; зниження вартості виробництва, за рахунок коректного визначення необхідної кількості функцій; маркетингова компанія
6. За інтенсивністю	Марочна	Використання марочної стратегії укріплення на ринку. Створення бренду та його маркетинг.

5.1.4 Детальний аналіз умов конкуренції в галузі

Таблиця 5.9 – Аналіз конкуренції

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Висновки	Конкуренція висока проте залежить від функцій бота	Потенційних конкурентів, найближчим часом, немає	Сервіси хостингу програм можуть тимчасово бути недоступними	Необхідність надавати широкий спектр функцій	Існують декілька ботів, які разом надають схожі функції

Допоки бот забезпечує власну унікальність, шляхом надання більшого спектру функцій та кращої взаємодії із користувачем, можливості роботи на ринку високі. Необхідно також врахувати особливості сервісів хостингу та підібрати оптимальний.

5.1.5 Фактори конкурентоспроможності

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування*
1	Доступність	Необхідна можливість для користувача отримати доступ до бота будь-коли
2	Функціонал	Надання широкого функціоналу, враховуючи при цьому витрати на розробку
3	Надання інформацію про країни та міста	Для забезпечення мінімально необхідної функціональності, бот повинен мати таку функцію
4	Надання інформацію про страви	Для забезпечення мінімально необхідної функціональності, бот повинен мати таку функцію
5	Надання інформацію про готелі	Для забезпечення мінімально необхідної функціональності, бот повинен мати таку функцію
6	Надання інформацію про музику	Для забезпечення мінімально необхідної функціональності, бот повинен мати таку функцію
7	Виділення ключових слів із повідомлення	Значно покращить досвід користувача

5.1.6 Аналіз сильних та слабких сторін стартап-проекту

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з моїм продуктом*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Доступність	20				+			
2	Функціонал	15					+		
3	Надання інформацію про країни та міста	5				+			
4	Надання інформацію про страви	5				+			
5	Надання інформацію про готелі	5				+			
6	Надання інформацію про музику	5			+				
7	Виділення ключових слів із повідомлення	20	+						

Аналізуючи попередню таблицю виділимо важливість такої функції бота, як виділення ключових слів із повідомлення. Оскільки вона є ключовим фактором диференціації мого продукту від конкурентів, то необхідно прикласти максимальні зусилля для її, якомога кращої, реалізації. Не слід також забувати, що хоч і для кожної окремої функції є певна альтернатива, не існує бота який реалізував би їх усіх.

5.1.7 SWOT-аналіз

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони: - Сукупність функцій - Виділення ключових слів із повідомлення	Слабкі сторони: - Поки мала аудиторія
Можливості: - Маркетингова компанія для приваблення користувачів - Впровадження додаткових функцій	Загрози: - Мінливі вимоги користувачів - Велика кількість конкурентів

5.1.8 Альтернативи ринкової поведінки

Проаналізувавши таблицю №13 оберемо варіант часткової реалізації функціоналу з подальшим поступовим релізом нових функцій. Як базові необхідні функції визначимо можливість надання інформацію про країни та міста, надання інформацію про готелі та виділення ключових слів із повідомлення користувача. Якщо перші дві функції реалізувати відносно просто, то остання значно ускладнює розробку, проте переваги від її реалізації значно перевищують затрати ресурсів. Саме ця функція слугуватиме основним інструментом утримання користувача.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Реалізація максимальної кількості функцій	Низька	1 рік
2	Часткова реалізація функцій з поступовим релізом нових	Висока	3 місяці
3	Відмова від деяких функцій	Низька	2 місяці

5.2 Розроблення ринкової стратегії проекту

5.2.1 Визначення стратегії охоплення ринку

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Цільова група потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Молодь (16 - 25)	Висока	Високий	Висока	Середня
2	Середнього віку (25 - 40)	Середня	Середній	Висока	Низька

За результатами аналізу потенційних груп споживачів (сегментів) обрано молодь, як цільову групу, для якої пропонуватиметься товар

Тому оберемо стратегію концентрованого маркетингу, як стратегію охоплення ринку.

5.2.2 Базова стратегія розвитку

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції*	*Базова стратегія розвитку*
Часткова реалізація функцій з поступовим релізом нових	Стратегія концентрованого маркетингу	Виділення ключових слів із повідомлення; Функціонал	Стратегія диференціації

5.2.3 Вибір стратегії конкурентної поведінки

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопро-хідцем» на ринку?*	*Пошук нових споживачів, або забирання існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1	Так	Пошук нових	Так	Розширення первинного попиту

Стратегія розширення первинного попиту доцільна у даному випадку, оскільки нашій фірмі, як лідеру недоцільно розмінюватися на боротьбу з невеликими конкурентами. Можна отримати велику економічну віддачу від розширення первинного рівня попиту. В цьому випадку компанія займатиметься реалізацією заходів по формуванню попиту (навчанню споживачів користуванню товаром, формуванню регулярного попиту, збільшенню разового споживання), а також пропаганду нових напрямів застосувань існуючого товару (бота), шляхом введення нових функцій.

5.2.4 Розробка стратегії позиціонування

Стратегія позиціонування полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції*	*Вибір асоціацій*
- Зручний інтерфейс - Красивий дизайн*	Стратегія диференціації	- Виділення ключових слів із повідомлення - Функціонал	- Функціональність - Зручність - Швидкість

* Слід розуміти дизайн взаємодії з користувачем

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії.

Як базову стратегію розвитку, обрано стратегію диференціації. Як стратегію конкурентної поведінки, обрано стратегію розширення первинного попиту.

Проаналізовано ключові вимоги до проекту та обрано такі асоціації: функціональність, зручність, швидкість.

5.3 Розроблення маркетингової програми стартап-проекту

5.3.1 Формування маркетингової концепції товару

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами*
1	Зручний інтерфейс	Використання сучасного месенджера для взаємодії	Простота та зручність взаємодії. Можливе використання сленгу для молодшої аудиторії
2	Красивий дизайн*	Виділення ключових слів із повідомлення	Відсутність такої функції практично у всіх конкурентів
3	Функціонал	Надання розширеного функціоналу	Обмежений функціонал у конкурентів

* Слід розуміти дизайн взаємодії з користувачем

5.3.2 Трирівнева маркетингова модель товару

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Зручний інструмент тематичного пошуку інформації на базі платформи Telegram		
	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Функціональність 2. Зручність 3. Швидкість		

Продовження таблиці 5.19

Рівні товару	*Сутність та складові*
	Якість: бота розробляється відповідно до стандартів оформлення коду та принципів SOLID
	Пакування відсутнє
	Марка: OS Bot development; Назва товару: Personalized InfoBot
	До продажу: Стандартний функціонал
	Після продажу: Розширений функціонал
Захист від копіювання: Патентування; Обфускація коду; Робота розробників лише на техніці компанії	

5.3.3 Визначення цінових меж

Необхідно визначити фактори, якими потрібно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів. Аналіз проводиться експертним методом. Базовий функціонал бота надаватиметься безплатно.

Таблиця 5.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники, грн/рік	*Рівень цін на товари-аналоги, грн/рік*	*Рівень доходів цільової групи споживачів, грн/місяць*	*Верхня та нижня межі встановлення ціни на товар/послугу, грн/рік*
3000	5000	10000	0-2400

5.3.4 Визначення оптимальної системи збуту

Оскільки продуктом в даному випадку є програма, то канал збуту не потрібен. Доставка товару користувачу відбувається власними засобами, та не потребує великих затрат часу чи ресурсів.

Таблиця 5.21 -Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Вибір послуг на сайті, оплата, постачання послуг	–	Виробник - споживач	Web-сайт

5.3.5 Розроблення концепції маркетингових комунікацій

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	*Канали комунікацій, цільові клієнти*	*Ключові позиції*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
Використання мобільних пристроїв для отримання контенту	Месенджери та соціальні мережі	Зручність та функціональність продукту	Донести користувачу інформацію про переваги над конкурентами	Акцентувати увагу на асоціаціях про продукт: - Функціональність - Зручність - Швидкість

Висновки

Безсумнівно є можливість ринкової комерціалізації проекту. Наявний попит на нові функції ботів, та стрімко зростає використання месенджерів загалом. Рентабельність роботи на ринку висока.

Існують перспективи впровадження з огляду на потенційні групи клієнтів. Бар’єри входження на ринок відносно високі, через високу конкуренцію практично в усіх вікових грумах, проте використання стратегії диференціації дозволить залучити більше користувачів. Конкурентоспроможність проекту – висока.

Доцільно обрати впровадження через часткову реалізацію функцій проекту, з поступовим впровадженням нових.

Подальша імплементація проекту доцільна.

ВИСНОВКИ

Під час написання магістерської дисертації були розглянуто історію становлення месенджерів, як сучасного засобу спілкування. Детально розглянуто ботів – програми які дозволяють автоматично спілкуватися з користувачами. Описано загальні принципи розроблення сервісу для платформи Телеграм та описано процес розробки бота. При цьому акцент зроблено на підборі інструментів необхідних для ефективної та гнучкої розробки різнопланових сервісів, від найпростіших до складних. Досліджено можливості використання засобів нейронних мереж для покращення досвіду користувача при використанні ботів.

Для створення сервісу тематичного аналізу інформації для платформи Телеграм обрано найпопулярніші на сьогодні засоби та інструменти: мову програмування C# разом із середовищем розробки Visual Studio 2017 Community, використано бібліотеку Telegram.Bot, яка реалізує Bot API Телеграм, для хостингу бота обрано Microsoft Azure, як засіб контролю версій та як інструмент CI/CD обрано GitLab, а також використано Microsoft Cognitive Services, як приклад покращення досвіду користувача засобами нейронних мереж.

Проаналізовано вже існуючих ботів та наведено приклади їх використання. В результаті, було виявлено необхідність популяризації використання розпізнавання команд із повідомлень користувачів, при використанні ботів як засобів роботи із клієнтами. Було систематизовано та формалізовано алгоритм вибору інструментів розробки. Розглянута фінансова сторона питання та підібрано декілька можливих варіантів для команд різного масштабу.

Розглянуто декілька способів використання нейронних мереж при роботі із користувачем, до основних з яких належить застосування їх для аналізу запиту користувача та для слідкування за змістом діалогу. При цьому описано можливості використання як самописних нейронних мереж, так і уже готових бібліотек, заточених під специфічні потреби розробників. Зокрема, значна увага приділяється використанню частини Microsoft Cognitive Services, яка дозволяє знаходити ключові

слова у повідомленні користувача. Також розглядаються перспективи використання інших частин даної бібліотеки для розширення функціоналу бота.

На базі описаних принципів та із використанням описаних технологій було створено сервіс для спілкування із користувачем, який дозволяє виділяти ключові слова із його повідомлень та надавати по них розширену довідку. Ця розробка є доказом концепції такого підходу до створення ботів для месенджерів та ілюструє можливу швидкість та гнучкість розробки.

Розроблено стартап проект «Personalized InfoBot», в якому розглянуто фінансову сторону розробки. За результатами цього розділу доведено можливість ринкової комерціалізації проекту, а також виділяються деякі моменти, які забезпечать збільшення прибутку.

ПЕРЕЛІК ПОСИЛАНЬ

1. PLATO (комп'ютерна система) [Електронний ресурс] – Режим доступу до ресурса: https://uk.wikipedia.org/wiki/PLATO_(комп'ютерна_система).

2. ICQ: 20 Years Is No Limit! [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@Dimitryophoto/icq-20-years-is-no-limit-8734e1eea8ea.

3. Saying goodbye to AIM, the instant messenger that changed how we communicate [Електронний ресурс] – Режим доступу до ресурсу: https://www.vox.com/culture/2017/12/15/16780418/aim-aol-instant-messenger-shutdown-cultural-impact.

4. Facebook [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Facebook.

5. Скороход В. ВИЗНАЧЕННЯ ЗАСОБІВ РОЗРОБКИ ЧАТ-БОТА «ПОМІЧНИК АБІТУРІЄНТА» ДЛЯ СУЧАСНИХ МЕСЕНДЖЕРІВ [Електронний ресурс] / Володимир Скороход – Режим доступу до ресурсу: https://phm.kspu.kr.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/99-2017/komp-iuterni-nauky-ta-informatsiini-tekhnolohii/1118-vyznachennya-zasobiv-rozrobky-chat-bota-pomichnyk-abituriyenta-dlya-suchasnykh-mesendzheriv.html.

6. Most popular global mobile messenger apps as of April 2018, based on number of monthly active users (in millions) [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.

7. Number of monthly active WhatsApp users worldwide from April 2013 to December 2017 (in millions) [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/260819/number-of-monthly-active-whatsapp-users/.

8. Number of monthly active Telegram users worldwide from March 2014 to December 2017 (in millions) [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/234038/telegram-messenger-mau-users/.

9. Нелюдська допомога: як боти рятують від самотності та хвороб [Електронний ресурс] – Режим доступу до ресурсу: http://ms.detector.media/web/online_media/nelyudska_dopomoga_yak_boti_ryatuyut_vid_samotnosti_ta_khvorob/.

10. Смолинець О. Т. Можливість використання нейронних мереж у системах обміну миттєвими повідомленнями [Електронний ресурс] / Остап Тарасович Смолинець – Режим доступу до ресурсу: https://nauka-online.com/ua/publications/informatsionnye-tehnologii/2018/5/vozmozhnost-ispolzovaniya-nejronnyh-setej-v-sisteme-obmena-mgnovennymi-soobshheniyami/.

11. Chatbots With Machine Learning: Building Neural Conversational Agents [Електронний ресурс] – Режим доступу до ресурсу: https://dzone.com/articles/chatbots-with-machine-learning-building-neural-con.

12. Bot API 2.0: информация для разработчиков [Електронний ресурс] – Режим доступу до ресурсу: https://tlgrm.ru/docs/bots/2-0-intro.

13. How to make a responsive telegram bot [Електронний ресурс] – Режим доступу до ресурсу: https://www.sohamkamani.com/blog/2016/09/21/making-a-telegram-bot/.

14. Обробка природної мови [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Обробка_природної_мови.

15. Named-entity Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Named-entity_recognition.

16. Mobile Google Assistant Now Works With Chromecast, Android TV [Електронний ресурс] – Режим доступу до ресурсу: https://www.androidheadlines.com/2017/10/mobile-google-assistant-now-works-with-chromecast-android-tv.html.

17. V-Soft Consulting. Chatbots 101 [Електронний ресурс] / V-Soft Consulting – Режим доступу до ресурсу: https://cdn2.hubspot.net/hubfs/1629777/Chatbots%20101.pdf?t=1526138378633.

18. How Do Summary Bots Like Smmry Work [Електронний ресурс] – Режим доступу до ресурсу: https://www.reddit.com/r/MachineLearning/comments/3u72za/how_do_summary_bots_like_smmry_work/.

19. How to order food conveniently through Chatbot? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@MarutiTech/how-to-order-food-conveniently-through-chatbot-12f2cfa82f59.

20. Cómo 4 chatbots respondieron al azar & coma [Електронний ресурс] – Режим доступу до ресурсу: http://www.waxnasbc.com/como-4-chatbots-respondieron-al-azar-preguntas-independientes/.

21. Webhook [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Webhook.

22. Server-side architecture when bots invade [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@JonathanZWhite/server-side-infrastructure-when-bots-invade-a2252e9d4bc9.

23. A Modern Bot Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://guild.beach.io/t/a-modern-bot-architecture/84.

24. Mindhacking [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/owocki/mindhacking/blob/master/README.md.

25. Celebi Tutorial: Neural Networks and Pattern Recognition Using MATLAB [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.byclb.com/TR/Tutorials/neural_networks/ch1_1.htm (дата звернення 10.05.2018). – Назва з екрана.

26. Типові архітектури нейронних мереж [Електронний ресурс] – Режим доступу до ресурсу: https://studfiles.net/preview/5740125/page:4/.

27. Neural-Networks-part-2 [Електронний ресурс] – Режим доступу до ресурсу: http://www.marekrei.com/blog/neural-networks-part-2-the-neuron/.

28. Neural-Networks-part-3 [Електронний ресурс] – Режим доступу до ресурсу: http://www.marekrei.com/blog/neural-networks-part-3-network/.

29. Фанифатьева А. Д. АВТОМАТИЧЕСКИЙ АНАЛИЗ ТОНАЛЬНОСТИ РЕЦЕНЗИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ TENSORFLOW [Електронний ресурс] / Фанифатьева А. Д. – Режим доступу до ресурсу: http://library.eltech.ru/files/vkr/2017/bakalavri/3382/2017%D0%92%D0%9A%D0%A0338212%D0%A4%D0%90%D0%9D%D0%98%D0%A4%D0%90%D0%A2%D0%AC%D0%95%D0%92%D0%90.pdf.

30. Convolutional Neural Networks (CNNs / ConvNets) [Електронний ресурс]:[Веб-сайт] – Режим доступу: http://cs231n.github.io/neural-networks-1/ (дата звернення 10.05.2018) – Назва з екрана.

31. Making a telegram bot [Електронний ресурс] – Режим доступу до ресурсу: https://www.sohamkamani.com/blog/2016/09/21/making-a-telegram-bot/.

32. Bots: An introduction for developers [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots.

33. Прикладний програмний інтерфейс [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс.

34. .NET Framework [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/.NET_Framework.

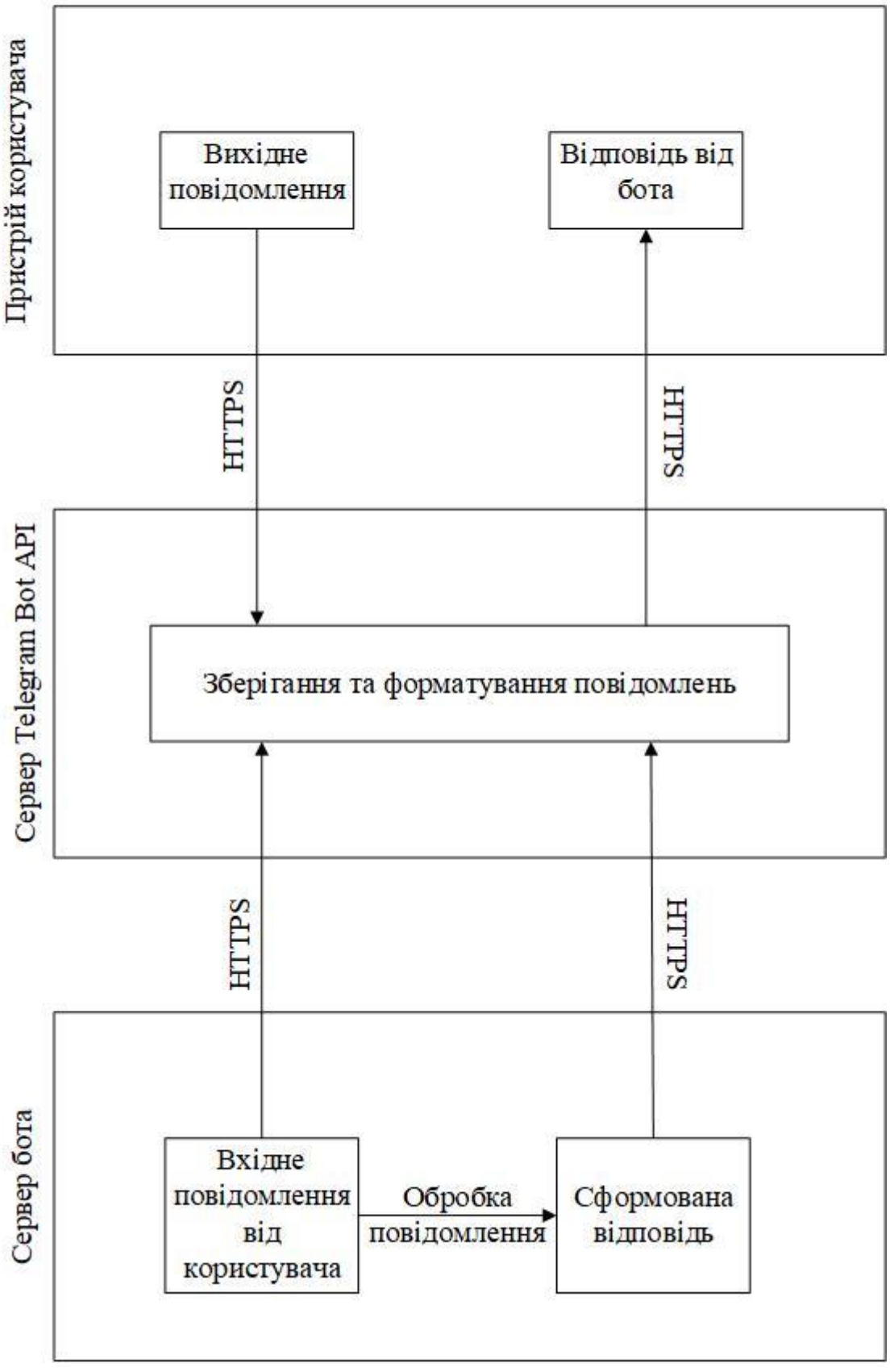
35. File:.NET Framework-Core relationship.svg [Електронний ресурс] – Режим доступу до ресурсу: https://en.m.wikipedia.org/wiki/File:.NET_Framework-Core_relationship.svg.

36. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com.

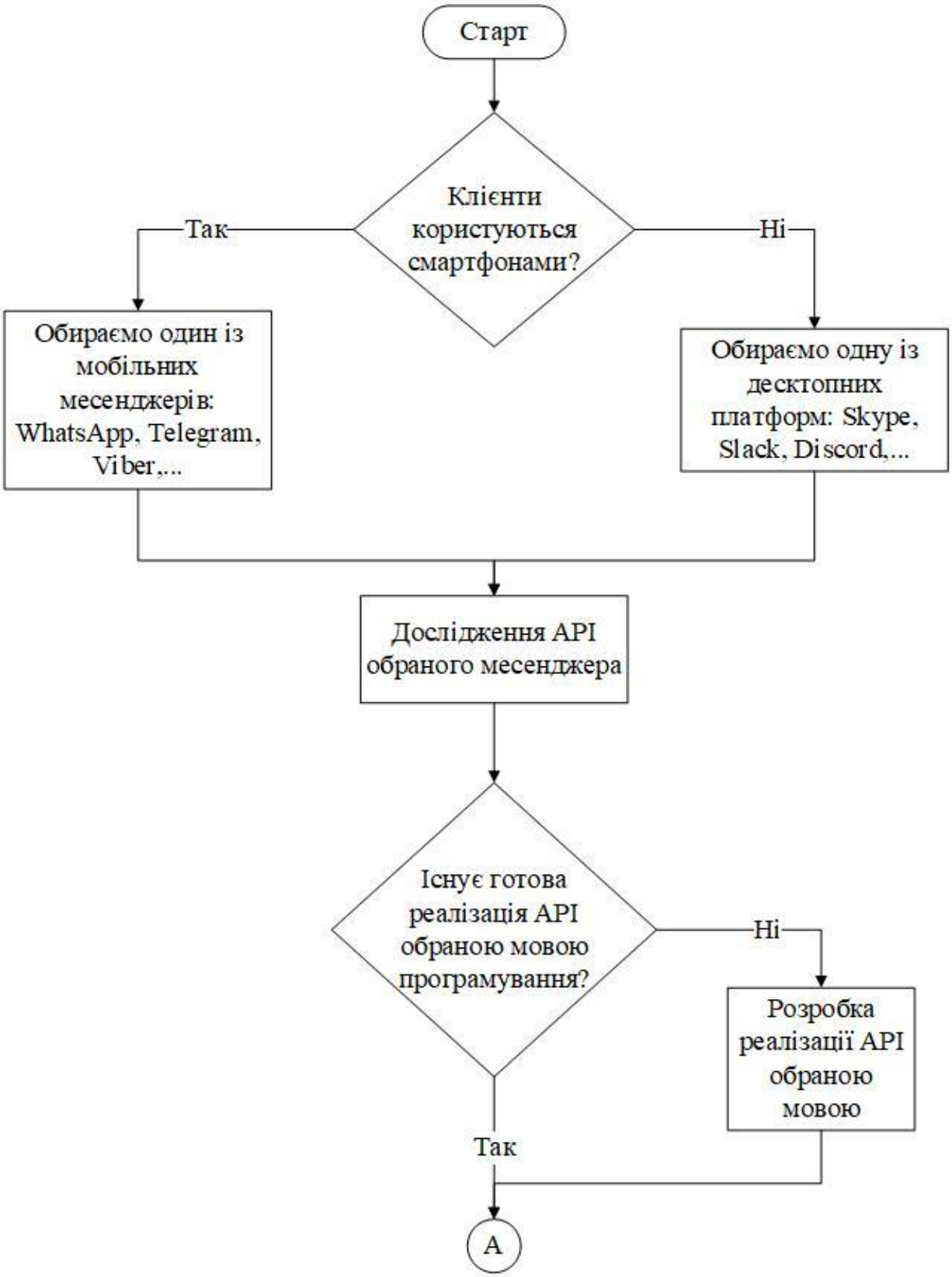
37. Telegram.Bot.Examples [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/TelegramBots/telegram.bot.examples.

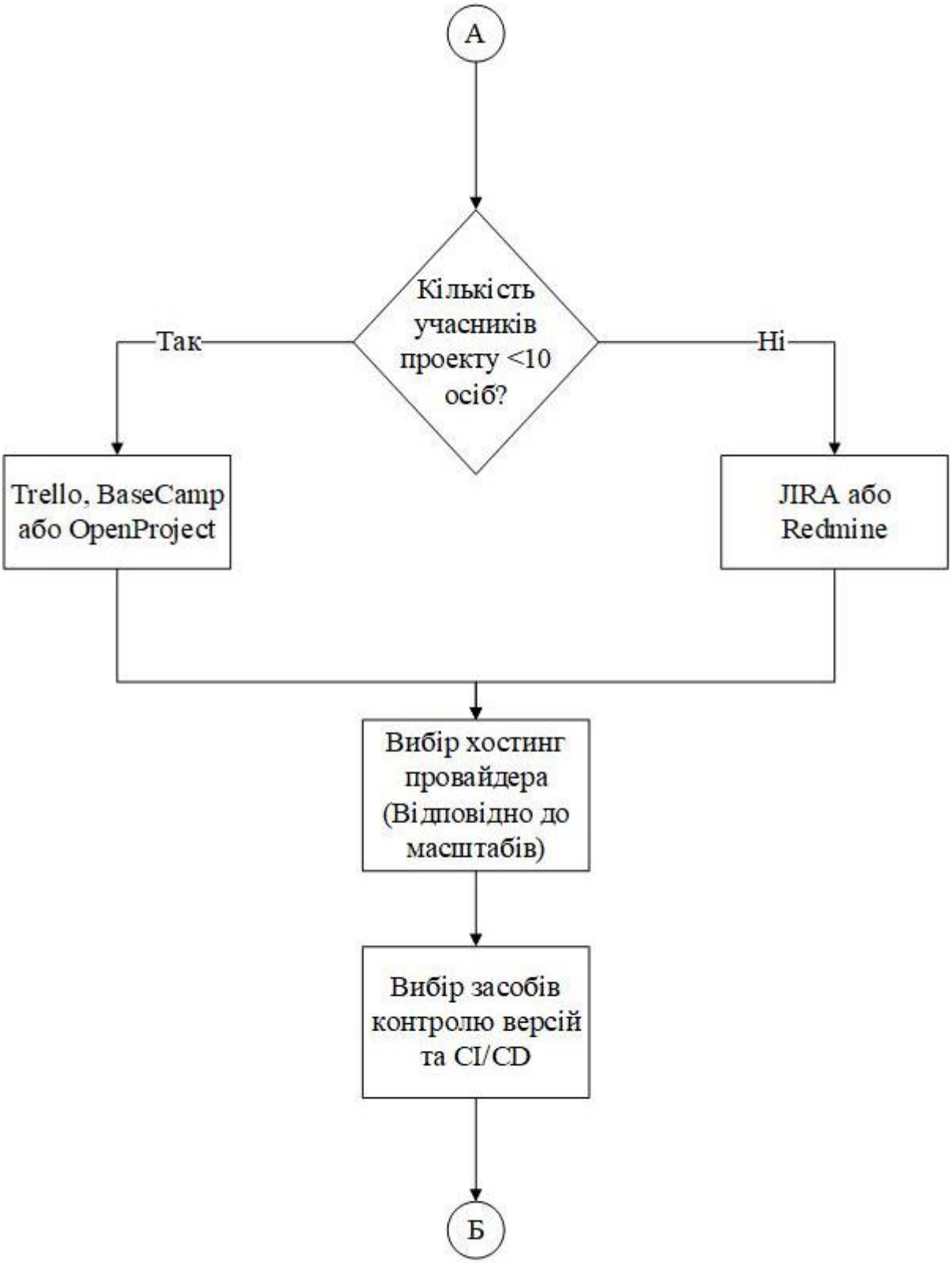
38. CI/CD: принципы, внедрение, инструменты [Електронний ресурс] – Режим доступу до ресурсу: https://goo.gl/624yfS.

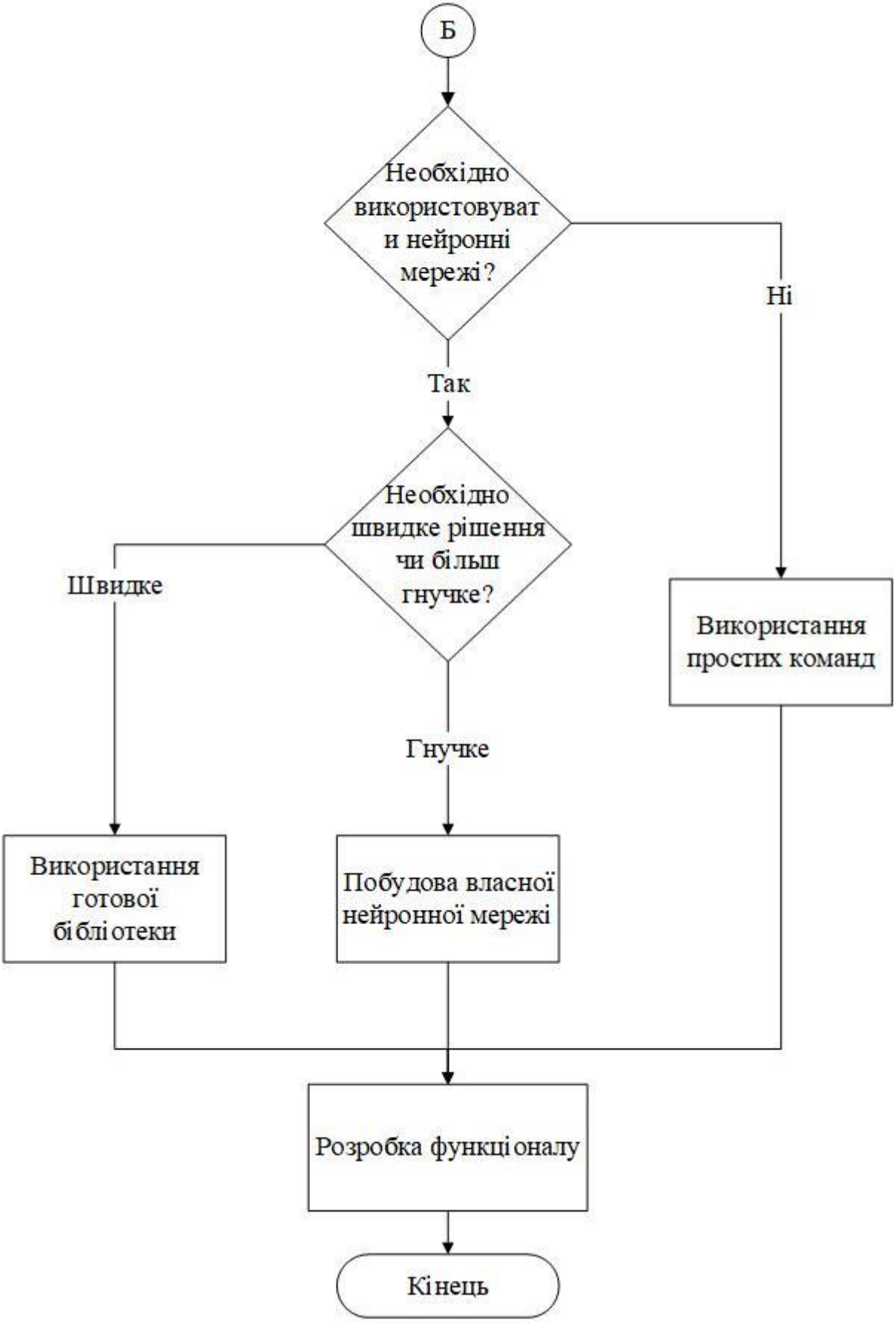
Додаток А
Взаємозв'язок елементів при використанні бота для СМОП



Додаток Б
Алгоритм створення бота




А
Кількість учасників проекту <10 осіб?
Так
Ні
Trello, BaseCamp або OpenProject
JIRA або Redmine
Вибір хостинг провайдера (Відповідно до масштабів)
Вибір засобів контролю версій та CI/CD
Б


Б
Необхідно використовувати нейронні мережі?
Ні
Так
Необхідно швидке рішення чи більш гнучке?
Швидке
Використання простих команд
Гнучке
Використання готової бібліотеки
Побудова власної нейронної мережі
Розробка функціоналу
Кінець

УДК 004.89

Смолинець Остап Тарасович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Смолинець Остап Тарасович
студент
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Smolynets Ostap
Student of the
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

МОЖЛИВІСТЬ ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ У СИСТЕМАХ ОБМІНУ МИТТЄВИМИ ПОВІДОМЛЕННЯМИ

ВОЗМОЖНОСТЬ ИСПОЛЬЗОВАНИЯ НЕЙРОННЫХ СЕТЕЙ В СИСТЕМЕ ОБМЕНА МГНОВЕННЫМИ СООБЩЕНИЯМИ

POSSIBILITIES OF USING NEURAL NETWORKS IN INSTANT MESSAGING SYSTEMS

Анотація. *У дослідженні здійснено аналіз взаємодії користувача з існуючими системами обміну миттєвими повідомленнями. Розглянуто ключові фактори взаємодії. Проведено огляд можливостей застосування нейронних мереж для покращення взаємодії користувача з системою.*

Ключові слова: *система обміну миттєвими повідомленнями, месенджер, нейронна мережа, досвід користувача.*

Аннотация. *В исследовании проведен анализ взаимодействия пользователя с существующими системами обмена мгновенными сообщениями. Рассмотрены ключевые факторы взаимодействия. Проведен обзор возможностей применения нейронных сетей для улучшения взаимодействия пользователя с системой.*

Ключевые слова: *система обмена мгновенными сообщениями, мессенджер, нейронная сеть, опыт пользователя.*

Summary. *The research carried out an analysis of user interaction with existing instant messaging systems. Key factors of interaction are considered. The use of neural networks to improve user interaction with the system is overviewed.*

Key words: *instant messaging system, messenger, neural network, user experience.*

Постановка проблеми. Системи обміну миттєвими повідомленнями (месенджери) користуються популярністю як серед підлітків, так і у старшого покоління. Вони відіграють ключову роль у спілкуванні між людьми, а також слугують засобом отримання інформації різної тематики для багатьох користувачів. Зокрема одним із засобів отримання інформації може слугувати застосунок інтегрований у дану систему у вигляді бота. Важливою частиною проектування таких застосунків для даних систем є власне досвід користувача при взаємодії із ними. Оскільки кількість користувачів та їхня активність є визначниками успішності застосунку слід максимізувати позитивний досвід користувача.

Аналіз останніх досліджень і публікацій. Серед актуальних матеріалів на дану тему можна виділити статтю авторства Nicole Radziwill (Ніколь Радзівілл) та Morgan Benton (Морган Бентон).[1] У даній статті розглядаються загальні принципи та методи оцінки якості розробки та впровадження сучасних чат-ботів. Зокрема надаються приклади так званого маскування ботів під справжніх людей та небезпеки пов’язані з цим.

Щодо статтей на вітчизняному просторі вартою уваги є стаття Скорохода Володимира. [2] У даній статті автор аналізує можливості створення чат-ботів у

сучасних месенджерах та визначає зручний у його випадку фреймворк, API та бібліотеки, для реалізації власного чат-бота.

Метою даної роботи є огляд основних принципів взаємодії користувача з системами обміну миттєвими повідомленнями та дослідження можливостей поліпшення даного досвіду за допомогою використання нейронних мереж.

Виклад основного матеріалу. Останні декілька років популярність систем обміну миттєвими повідомленнями (месенджерів) тільки зростає. Вони перетворилися із засобів для спілкування між людьми у засоби для отримання інформації та у неймовірно потужний маркетинговий інструмент.

Для початку необхідно розглянути історію популярності засобів віддаленого спілкування між людьми. У давні часи єдиним способом віддаленого спілкування були листи. Очевидно що даний спосіб був не тільки повільним, але й надзвичайно ненадійним. Далі значну роль у віддаленому спілкуванні мав телеграф, який у 19 столітті дозволив передавати кодовані повідомлення на великі відстані. Та попри широке застосування він був також ненадійним та незручним.

Наступним проривом стало винайдення телефону. Перший прототип запатентовано в 1876 році американським винахідником Александром Беллом. Безсумнівно телефон став новим засобом зручного та швидкого зв’язку на відстані та дотепер є найрозповсюдженішою мережею передачі інформації. Телефонний зв'язок за історію свого існування пройшов шлях від виключно провідного до безпровідного засобу зв’язку, від пристрою єдиною метою якого було зв’язати двох користувачів до фактично портативного комп’ютера. Не варто також забувати про можливості відеозв’язку.

Останні декілька років традиційні телефонні дзвінки стають усе менш популярними у порівнянні із засобами миттєвого обміну повідомленнями чи сервісами онлайн зв’язку. Більше того багато людей та компаній надають перевагу текстовим чи мультимедійним повідомленням у порівнянні із дзвінками. Інтернет

сервісами для спілкування також прогресують разом із технологіями - спочатку популярними були чати, потім месенджери, потім соціальні мережі, а тепер перспективними знову вважаються месенджери. Причина повторної хвилі їх популярності – зміни в області мобільного Інтернету: високі швидкості, нижчі ціни, та широке поширення смартфонів.[2]

На графіку на Рис.1 представлено кількість активних користувачів у найпопулярніших месенджерах станом на квітень 2018 року. Як бачимо очевидним лідером є WhatsApp із загальною кількістю користувачів у приблизно 1.5 мільярди. Також варто звернути увагу на Telegram, показники якого подвоїлися у порівнянні із жовтнем 2017 року.

Очевидно, що всі месенджери у даному списку різні, проте у них є спільні риси, які стали причинами їхнього успіху. Практично у кожного із них зручний, сучасний та зрозумілий інтерфейс користувача. Також кожен із месенджерів підтримує групові чати. Деякі з них підтримують канали – частково повторюють функціонал чатів, за винятком того що можливість публікувати повідомлення мають лише користувачі із відповідними правами доступу.

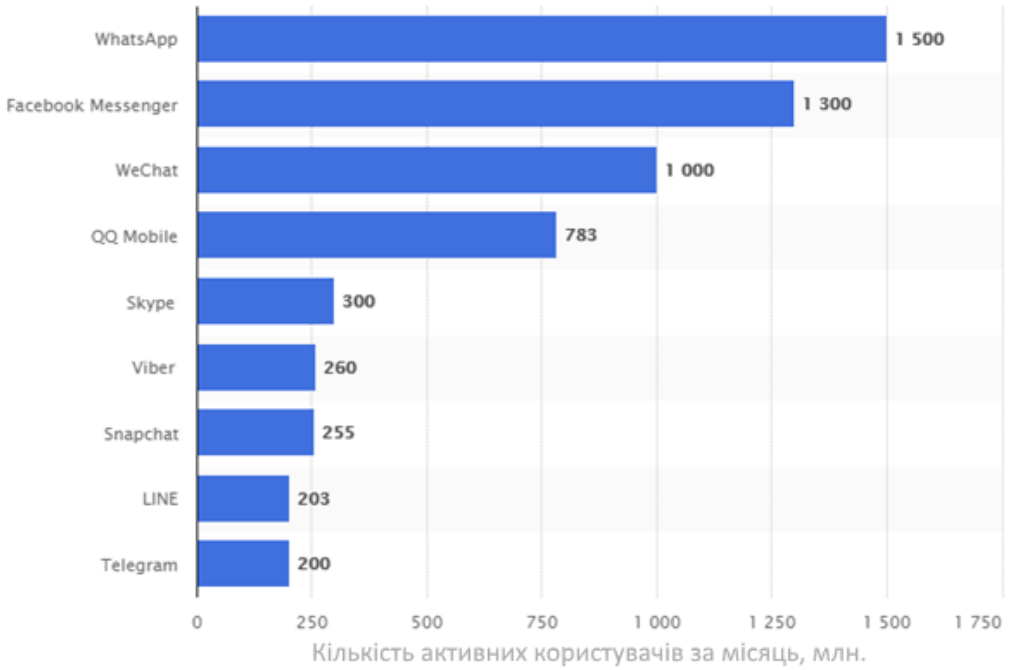


Рис. 1. Кількість активних користувачів у найпопулярніших месенджерах [3]

Ще одною особливістю є те, що усі наведені вище месенджери, так чи інакше, підтримують інтеграцію із програмами-ботами. Snapchat та QQ Mobile підтримують так звану сіру інтеграцію – офіційно вони не підтримують використання ботів, та не регулюють використання сервісів програмами під виглядом користувачів. Усі ж інші підтримують інтеграцію ботів на рівні платформи.

Чат-ботів можна поділити на ботів для розмов на широкий спектр тем та орієнтованих на мету. Перші призначені для діалогу із користувачем без певної мети, в той час як другі – орієнтовані на вирішення щоденних проблем засобами «природньої» мови. Боти орієнтовані на допомогу користувачу у досягненні певної мети (регулярне отримання корисної інформації, встановлення нагадувань, тощо) є найбільш поширеними та саме про них йтиметься далі у статті.

Як і при роботі із програмою, при використанні бота важливий інтерфейс взаємодії із користувачем. Інтерфейс власне месенджера чітко визначений, проте інтерфейс бота це поняття, в певному сенсі більш розмите.

До взаємодії із ботом можна віднести підтримку команд, можливості виокремлення команд із повідомлення користувача та можливість розуміти контекст діалогу. Підтримка команд – це те, що відрізняє бота від звичайного користувача. Більшість ботів обмежуються цим функціоналом. До прикладу при наявності такого функціоналу можна реалізувати таку команду: *«Будь ласка, установи таймер на 10:00».* Виокремлення команд із повідомлення користувача значно підвищує привабливість бота, а тому стає все популярнішою функцією. До прикладу при наявності такого функціоналу бот розумітиме різноманітні варіації попередньої команди, наприклад: *«Чи не міг би ти установити таймер на 10:00 26-го червня»,* що є дуже схожим на «природню» розмову. Для прикладу можливості розуміти контекст діалогу наведемо діалог (Л – людина, Б- бот):

Л – Додай банани у список покупок.

Б – Звісно.

Л – Також потрібно купити упаковку борошна.

Б – Додано.

Результатом стане список покупок, що міститиме і банани, і упаковку борошна. У даному випадку, для гнучкості розробки бота та для забезпечення можливості зрозуміти зв’язаність повідомлень, застосовуються нейронні мережі.

Схожим застосуванням є використання нейронних мереж для персоналізації та вдосконалення виводу інформації користувачеві. Наприклад, бот проаналізував недавнє спілкування та прийшов до висновку що користувач надає перевагу процесорам марки Intel, тому на запит «Я хотів би купити процесор» він відповість інформацією про продукцію даної марки.

Висновки. Сучасний користувач надає перевагу використанню програм-месенджерів над традиційними дзвінками. При цьому невід’ємною частиною взаємодії із месенджерами є використання ботів. Оскільки ключовим для розробника є кінцевий досвід експлуатації користувачем продукту, слід при розробці бота врахувати сучасні тенденції та максимізувати при цьому позитивний досвід. Отже, необхідно використовувати нейронні мережі для забезпечення максимальної функціональності та відповідно максимального задоволення користувача.

Література

1. Radziwill N. Evaluating Quality of Chatbots and Intelligent Conversational Agents [Електронний ресурс] / N. Radziwill, M. Benton. – 2017. – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1704/1704.04579.pdf.
2. Скороход В. ВИЗНАЧЕННЯ ЗАСОБІВ РОЗРОБКИ ЧАТ-БОТА «ПОМІЧНИК АБІТУРІЄНТА» ДЛЯ СУЧАСНИХ МЕСЕНДЖЕРІВ [Електронний ресурс] / Володимир Скороход. – 2017. – Режим доступу до ресурсу: https://phm.kspu.kr.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/99-2017/komp-iuterni-nauky-ta-informatsiini-tekhnolohii/1118-vyznachennya-zasobiv-rozrobky-chat-bota-pomichnyk-abituriyenta-dlya-suchasnykh-mesendzheriv.html.
3. Most popular global mobile messenger apps [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.